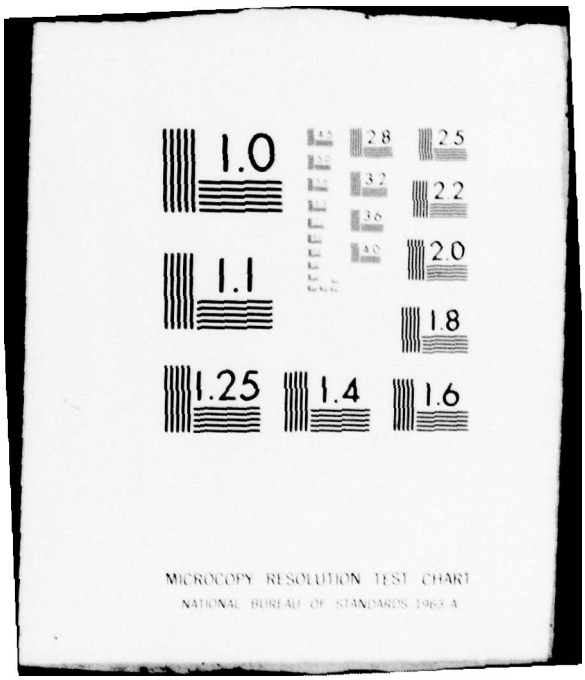
AD-A074 548 NAVAL UNDERWATER SYSTEMS CENTER NEW LONDON CT NEW LO--ETC F/G 12/1
GENERALIZED ADAMS METHODS.(U)
JUL 79 D LEE
UNCLASSIFIED NUSC-TR-6011 NL

| OF |
AD A074548

END
DATE FILMED
10-79
DDC


1.0
1.1
1.25
1.4
1.6
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.8
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

NUSC Technical Report 6011

AD A074548

DDC FILE COPY

(12) (9) NUSC Technical Report 6011

LEVEL II

DDC RECEIVED OCT 3 1979 E

(6) # Generalized Adams Methods

(14) NUSC-TR-6011

(10) Ding Lee
Special Projects Department

(12) 51

(16) ZR0000

(17) ZR000001

(11) 6 July 1979

NUSC

Naval Underwater Systems Center
Newport, Rhode Island • New London, Connecticut

405 918

79 10 02 061

PREFACE

This report was prepared under NUSC Project No. A65020, "Finite-Difference Solutions to Acoustic Wave Propagation," Principal Investigator, Dr. D. Lee (Code 312); Program Element 61152N, Navy Subproject/Task ZR0000101, "In-House Laboratory Independent Research," Program Manager, Dr. J. H. Probus (NAVMAT, Code 08T1).

The Technical Reviewer for this report was Dr. P. B. Abraham (Code 313).

The author wishes to thank Professor Stanley Preiser of the Polytechnic Institute of New York for evaluating the author's methods on the IBM 360/65 and validating his computations, which had been performed on PDP 11/70/.

Some sections of this report were presented at the SIAM 1978 Fall Meeting, which was held October 30 to November 1, 1978, in Knoxville, Tennessee, U.S.A.

REVIEWED AND APPROVED: 6 July 1979

W. A. VonWinkle
Associate Technical Director
for Technology

The author of this report is located at the New London Laboratory, Naval Underwater Systems Center, New London, Connecticut 06320.

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER TR 6011	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) GENERALIZED ADAMS METHODS		5. TYPE OF REPORT & PERIOD COVERED 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Ding Lee		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Underwater Systems Center New London Laboratory New London, CT 06320		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS A65020
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Material Command (Code 08T1) Washington, DC 20362		12. REPORT DATE 6 July 1979 13. NUMBER OF PAGES 45
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

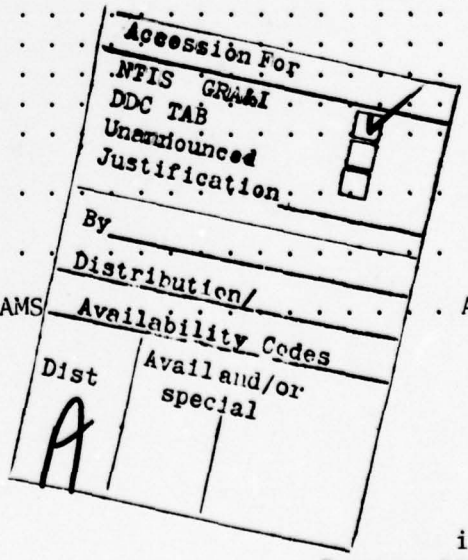
Generalized Adams-Bashforth (GAB) Method
Generalized Adams-Moulton (GAM) Method
Nonlinear Multistep (NLMS) Methods
Stiff Equations
Underwater Acoustic Wave Propagation

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

It is most desirable to use an accurate numerical method that allows the use of a large step size to solve stiff initial value problems. A family of generalized Adams-Bashforth and generalized Adams-Moulton methods, which we call GAB-GAM, are shown to be derived from nonlinear multistep (NLMS) methods. GAB-GAM methods are found to have advantages in solving (1) stiff equations and (2) a class of underwater acoustic wave propagation problems. An application of these methods is included in this report. Also, an ANSI FORTRAN program, together with a set of numerical test examples, is included.

DD FORM 1 JAN 73 1473

TABLE OF CONTENTS



GENERALIZED ADAMS METHODS

1. INTRODUCTION

Among linear multistep (LMS) methods, the Adams family is found to be the most effective,[1] to this date, for solving initial value problems in ordinary differential equations (ODE's). Adams methods are impractical in solving "stiff" equations because prohibitively small step sizes are required for accuracy. To achieve equal effectiveness of Adams methods for solving stiff equations, efforts were made by Certaine,[2] who suggested a generalization of Adams methods for step number $k = 1,2$; by Bjurel,[3] who modified Adams methods; by Norsett,[4] who gave an A-stable modification of Adams-Bashforth methods; and by Verwer,[5] who generalized LMS methods with zero-parasitic and an adaptive principal root. Jain[6] developed A-stable methods by means of Hermite interpolation polynomials, and Murphy[7] employed Newton's divided difference representation of the Hermite interpolation polynomials to develop a family of A-stable methods. Lambert[8] introduced multistep methods with variable matrix coefficients.

This report shows that an appropriate choice of the characteristic polynomial coefficients of the nonlinear multistep (NLMS) methods yields the generalized Adams-Bashforth and generalized Adams-Moulton methods (GAB-GAM). These methods are shown to have portions in common with the methods of Certaine, Bjurel, Norsett, Jain, Murphy, Verwer, and Lambert. An examination of how NLMS methods yield GAB-GAM follows. A section of theory, describing the consistency, stability, and convergence of GAB-GAM methods, will be outlined. In addition, the property of A-stability of GAB-GAM will be outlined. GAB-GAM can solve stiff equations effectively. However, this is not a restriction; GAB-GAM can be used to solve nonstiff equations as well, although this is inefficient for that class of problems.

We have included here an application of GAB-GAM to a class of underwater acoustic parabolic wave equations. A package of ANSI FORTRAN (GABM) programs, designed to implement GAB-GAM methods, is introduced. The important features of the GABM package are discussed in a separate section. Test results of GABM are compared with a set of previously published results by other techniques. Computations were performed on a PDP-11/70 computer, using double precision arithmetic. A listing of ANSI FORTRAN (GABM) programs is included in the appendix.

2. FORMULATION

Initial value problems of ODE's, generally, can be expressed by

$$y' = f(t,y);\ y(t_0) = y_0 \ . \tag{2-1}$$

A class of systems of first order ODE's, called stiff, arises in the applications of chemical kinetics, reactor kinetics, missile guidance, . . ., etc., and often takes the form

$$y' = Ay + g(t,y);\ y(t_0) = y_0 \ , \tag{2-2}$$

where $Re\{\lambda(A)\} < 0$ and $\lambda(A)$, the eigenvalues of A, differ greatly in magnitude. We choose to deal with equation (2-2), since the applications frequently appear in this form.

In solving equation (2-1), k-step LMS methods[9] can be applied that take the form

$$\sum_{i=0}^{k} \alpha_i y_{n+i} = h \sum_{i=0}^{k} \beta_i f_{n+i} \ , \tag{2-3}$$

where $\alpha_k \neq 0$, $|\alpha_0| + |\beta_0| > 0$.

LMS can solve equation (2-1) effectively when $||\partial f/\partial y||$, the norm of the Jacobian matrix, is small. Our approach is to generalize equation (2-3) such that when $||\partial f/\partial y||$ is large LMS can be used to solve equation (2-2) effectively. This led to what we refer to as NLMS methods[10] that possess the form

$$\sum_{i=0}^{k} \alpha_i e^{Ah(k-i)} y_{n+i} = h \sum_{i=0}^{k} \phi_{ki}(Ah) g_{n+i} \ , \tag{2-4}$$

where $\alpha_k \neq 0$, $|\alpha_0| + |\lambda(\phi_{k0}(Ah))| > 0$.

In carrying out the complete development, A is assumed to be a constant nonsingular matrix. Methods to handle A(t) will be explained in a later section. It can be seen easily that when A = 0, equation (2-2) reduces to equation (2-1) and equation (2-4) reduces to equation (2-3), where $\lim_{||A|| \to 0} \phi_{ki}(0) = \beta_i I$, where I is the identity matrix. This

shows that NLMS methods include LMS as a subset. The theory of the NLMS methods also generalizes the theory of the LMS methods, as has been demonstrated in reference 10.

The complete formulation of NLMS involves the accurate determination of $\phi_{ki}(Ah)$ with the selection of α_i. This appears in its entirety in references 10 and 11. A brief outline of NLMS methods is given below.

We first define the NLMS operator,

$$L[y(t);h] = \sum_{i=0}^{k} \alpha_i e^{Ah(k-i)} y(t + ih) - h \sum_{i=0}^{k} \phi_{ki}(Ah) g(t + ih, y(t + ih)) . \tag{2-5}$$

Then, for a constant matrix A, we express equation (2-2) by

$$\frac{d}{dt}\left(e^{-At}y\right) = e^{-At}g(t,y),\ y(t_0) = y_0 \tag{2-6}$$

and integrate equation (2-6) over the interval $[t_n, t_{n+i}]$ to obtain

$$y(t_{n+i}) = e^{iAh}y(t_n) + \int_{t_n}^{t_{n+i}} e^{A(t_{n+i}-t')}g(t',y)dt' . \tag{2-7}$$

If we expand $y(t_n + ih)$ and $g(t',y)$ of equation (2-7) in a Taylor expansion around t_n, substitute into equation (2-5), and simplify, we obtain

$$L[y(t);h] = \left\{ \sum_{i=0}^{k} \alpha_i e^{Ah(k-i)} e^{iAh} y \right\} + \sum_{j=0}^{\infty} C_j(Ah) g^{(j)} , \tag{2-8}$$

where

$$C_j(Ah) = \sum_{i=0}^{k} \frac{\alpha_i}{j!} e^{Ah} \left[\int_{t_n}^{t_{n+i}} e^{Ah(t_{n+i}-t')} (t' - t_n)^j dt' \right] - h \sum_{i=0}^{k} \frac{(ih)^j}{j!} \phi_{ki}(Ah) \quad . \tag{2-9}$$

In view of the first condition of consistency,[10]

$$\sum_{i=0}^{k} \alpha_i = 0 \ ,$$

which implies that { } of equation (2-8) is equal to zero, we obtain

$$L[y(t);h] = \sum_{j=0}^{\infty} C_j(Ah) g^{(j)} \quad . \tag{2-10}$$

Since A is assumed to be a constant nonsingular matrix, we can multiply both sides of equation (2-9) by A^{j+1}, and we obtain

$$\frac{A^{j+1}}{j!} \int_{t_n}^{t_{n+i}} e^{Ah(t_{n+i}-t')} (t' - t_n)^j dt' = e^{iAh} - \sum_{\ell=0}^{j} \frac{(iAh)^\ell}{\ell!} . \tag{2-11}$$

Using equation (2-11) and simplifying equation (2-9), we obtain

$$-A^{j+1} C_j(Ah) = \sum_{i=0}^{k} \alpha_i e^{Ah(k-i)} \sum_{\ell=0}^{j} \frac{(iAh)^\ell}{\ell!} + \frac{(Ah)^{j+i}}{j!} \sum_{i=0}^{k} (i)^j \phi_{ki}(Ah) \quad . \tag{2-12}$$

We require that $C_j(Ah) = 0$ for $j = 0,1,2,\ldots,p$, but $C_{p+1}(Ah) \neq 0$. This requirement defines an NLMS method of order p. The usual

consistency conditions for LMS are generalized.[10] This permits formulating NLMS methods in a matrix form, leading to the solution of $\phi_{ki}(Ah)$ explicitly when the characteristic polynomial coefficients are selected. In requiring that $C_j(Ah) = 0$ for $j = 0,1,2,\ldots,p$, we also select the order p to coincide with the step number k; this selection ensures a unique solution so that $\phi_{ki}(Ah)$ can be solved explicitly.

We now select the characteristic polynomial coefficients $\alpha_k = 1$, $\alpha_{k-1} = -1$, $\alpha_{k-2} = \alpha_{k-3} = \ldots = \alpha_0 = 0$. This selection leads to the formulation of Adams and generalized Adams methods.

For a constant nonsingular A, we multiply equation (2-12) by $(Ah)^{-(j+1)}$, let $||A|| \to 0$ for $j = 0,1,2,\ldots,p-1$, and define $\phi_{ki}(0) = \beta_i$ as

$$\sum_{i=0}^{k} \alpha_i \frac{i^{j+1}}{(j+1)!} = \frac{1}{j!} \sum_{i=0}^{k} i^j \beta_i \ . \qquad (2\text{-}13)$$

Equation (2-13) relates β_i to α_i and gives the second condition of consistency of LMS methods.

When $\phi_{kk}(Ah) = 0$, equation (2-12) is expressed as a predictor, as follows:

$$\begin{bmatrix} I & I \\ I + (k-1)Ah & I + kAh \\ \vdots & \vdots \\ \sum_{m=0}^{p-1} \frac{[(k-1)Ah]^m}{m!} & \sum_{m=0}^{p-1} \frac{(kAh)^m}{m!} \end{bmatrix} \begin{bmatrix} -e^{Ah} \\ I \end{bmatrix} = \qquad (2\text{-}14)$$

$$\begin{bmatrix} \frac{Ah}{0!} & & \bigcirc \\ & \frac{(Ah)^2}{1!} & \\ \bigcirc & & \frac{(Ah)^p}{(p-1)!} \end{bmatrix} \begin{bmatrix} I & I & \cdots & I \\ 0 & I & \cdots & (k-1)I \\ & & \vdots & \\ 0 & I & \cdots & (k-1)^{p-1}I \end{bmatrix} \begin{bmatrix} \phi_{k0}(Ah) \\ \phi_{k1}(Ah) \\ \vdots \\ \phi_{k,k-1}(Ah) \end{bmatrix}.$$

We can find $\phi_{ki}(Ah)$ as follows:

Step k	$\phi_{k0}(Ah)$, $\phi_{k1}(Ah)$, $\cdots$, $\phi_{k,k-1}(Ah)$
1	$-(Ah)^{-1}(I - e^{Ah})$
2	$-(Ah)^{-2}[e^{Ah} - (I + Ah)]$, $-(Ah)^{-2}[-(I + Ah)e^{Ah} + (I + 2Ah)]$
3	$-(Ah)^{-3}[-(I + \frac{Ah}{2})e^{Ah} + (I + \frac{3}{2}Ah + (Ah)^2)]$, $-(Ah)^{-3}[2(I + Ah)e^{Ah} - (2I + 4Ah + 3(Ah)^2)]$, $-(Ah)^{-3}[-(I + \frac{3}{2}Ah + (Ah)^2)e^{Ah} + (I + \frac{5}{2}Ah + 3(Ah)^2)]$.
.	.
.	.
.	.

(2-15)

Substituting $\phi_{ki}(Ah)$ of equation (2-14), we have the Generalized Adams-Bashforth (GAB) method.

When $\phi_{kk}(Ah) \neq 0$, equation (2-12) is expressed as a corrector, as follows:

$$\begin{bmatrix} I & I \\ I + (k-1)Ah & I + kAh \\ & \vdots \\ \sum_{m=0}^{p} \frac{((k-1)Ah)^m}{m!} & \sum_{m=0}^{p} \frac{(kAh)^m}{m!} \end{bmatrix} \begin{bmatrix} -e^{Ah} \\ I \end{bmatrix} = \tag{2-16}$$

$$\begin{bmatrix} \frac{Ah}{0!} & & \bigcirc \\ & \frac{(Ah)^2}{1!} & \\ \bigcirc & & \frac{(Ah)^{p+1}}{p!} \end{bmatrix} \begin{bmatrix} I & I & \cdots & I \\ 0 & I & \cdots & kI \\ & & \vdots & \\ 0 & I & \cdots & k^pI \end{bmatrix} \begin{bmatrix} \phi_{k0}(Ah) \\ \phi_{k1}(Ah) \\ \vdots \\ \phi_{kk}(Ah) \end{bmatrix} .$$

Similarly, we can find the $\phi_{ki}(Ah)$ as follows:

Step k	$\phi_{k0}(Ah), \cdots, \phi_{kk}(Ah)$
1	$-(Ah)^{-2}[-(Ah - I)e^{Ah} - I], -(Ah)^{-2}[-e^{Ah} + (I + Ah)]$
2	$-(Ah)^{-3}[-(I - \frac{Ah}{2})e^{Ah} + (I + \frac{Ah}{2})]$, $-(Ah)^{-3}[(2I - (Ah)^2)e^{Ah} - 2(I + Ah)]$, $-(Ah)^{-3}[-(I + \frac{Ah}{2})e^{Ah} + (I + \frac{3}{2}Ah + (Ah)^2)]$.
3	$-(Ah)^{-4}[(I - \frac{1}{6}(Ah)^2)e^{Ah} - (I + Ah + \frac{1}{3}(Ah)^2)]$, $-(Ah)^{-4}[-(3I + Ah - (Ah)^2)e^{Ah} + (3I + 4Ah + \frac{3}{2}(Ah)^2)]$, $-(Ah)^{-4}[(3I + 2Ah - \frac{1}{2}(Ah)^2 - (Ah)^3)e^{Ah} - (3I + 5Ah + 3(Ah)^2)]$, $-(Ah)^{-4}[-(I + Ah + \frac{1}{3}(Ah)^2)e^{Ah} + (I + 2Ah + \frac{11}{6}(Ah)^2 + (Ah)^3)]$.
.	.
.	.
.	.

(2-17)

Substituting $\phi_{ki}(Ah)$ of equation (2-17) into equation (2-2), we have the Generalized Adams-Moulton (GAM) method. Simplifying the terms of $\phi_{ki}(Ah)$ of GAB-GAM and letting $||A|| \to 0$, one obtains the Adams-Bashforth and Adams-Moulton methods, respectively.

3. THEORY

Details of the complete theory of NLMS methods with regard to consistency, stability, and convergence can be found in reference 10. The theory of NLMS methods is automatically applicable to GAB-GAM methods. A sketch of the theory of GAB-GAM methods is given here.

CONSISTENCY

An NLMS method is defined to be of order p when $C_j(Ah)$ is selected to be 0 for j = 0,1, . . .,p, but $C_{p+1}(Ah) \neq 0$. This condition, $C_j(Ah) = 0$, implies that

$$\lim_{h\to 0} \left|\left| \sum_{i=0}^{k} \alpha_i e^{Ah(k-i)} y_{n+i} - h \sum_{i=0}^{k} \phi_{ki}(Ah) g_{n+i} \right|\right| = 0 . \qquad (3\text{-}1)$$

This equation yields the consistency for GAB-GAM methods.

If we let $||A|| \to 0$, we have shown that equation (2-12) reduces to

$$\sum_{i=0}^{k} \alpha_i \frac{i^{j+1}}{(j+1)!} = \frac{1}{j!} \sum_{i=0}^{k} i^j \beta_i . \qquad (3\text{-}2)$$

This is an alternate confirmation of the second condition of consistency for the LMS methods.

STABILITY

The characteristic polynomial of NLMS methods is defined by

$$\rho(\lambda,\zeta) = e^{Ah\lambda} \rho(\zeta) , \qquad (3\text{-}3)$$

where

$$\rho(\zeta) = \sum_{i=0}^{k} \alpha_i \zeta^i . \qquad (3\text{-}4)$$

In view of the choice of α_1, GAB-GAM are strongly stable. This choice also gives the first condition of consistency of both LMS and NLMS methods.

CONVERGENCE

Theorem

GAB-GAM methods are convergent.

This theorem has been proved in reference 10.

A-Stability

Dahlquist[12] defined a method to be A-stable if the numerical solution $||y_n|| \to 0$ asymptotically as $n \to \infty$ for the differential equation $y' = Ay$, where $\mathrm{Re}\{\lambda(A)\} < 0$. GAB-GAM methods have this property.

Theorem

GAB-GAM methods are A-stable in the sense of Dahlquist.

If we apply GAB-GAM methods to solve the problem $y' = Ay$, which implies $g(t,y) = 0$, the GAB-GAM methods produce the exact solution to the problem; i.e.,

$$y_n = e^{At_n} y_0 = e^{nAh} y_0 .$$

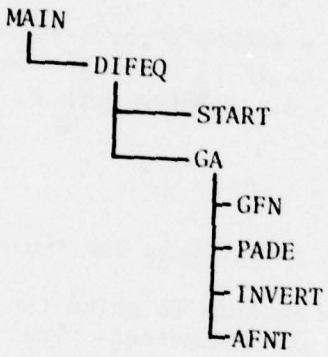
We calculate the e^{nAh} by Pade approximation. If $\mathrm{Re}\{\lambda(A)\} < 0$, then $\lim_{n\to\infty} ||y_n|| \to 0$, thus establishing the A-stability.

4. IMPLEMENTATION OF GAB-GAM METHODS

A package of computer programs was written in ANSI FORTRAN language. There are a number of useful features incorporated in the package, namely, variable-step-size, predict-and-correct-m-times, self-starting techniques, and the handling of variable and time-dependent A(t). The usage of this package is self-explanatory. A description of the package is given below.

PROGRAM STRUCTURE

This package consists of nine programs, of which three are user-supplied. The interrelationship among the programs is shown by the diagram below.



The main control program (MAIN) processes all the inputs and calls a secondary control subroutine (DIFEQ). DIFEQ sets up the iterative procedure; calls for the GAB-GAM methods; and controls the self-starter, step-size changes, predictor-and-corrector, and corrector's convergence, subject to the user-required tolerance. It also prepares the results for printout. The self-starter (START), GA, Pade approximation (PADE), matrix inversion (INVERT), and g(t,y) (GFN) are all written in subroutine formats. GFN must be supplied by the user.

FEATURES

1. Variable-Step Size (Fixed-Step Size as Well). In exercising the variable-step-size technique, if a change is needed, the step size is changed by halving or doubling, depending upon the need. The indicator IPC is used to indicate this step-size option. Setting IPC = 0 sets a fixed-step size, and IPC = 1 sets the use of variable-step size.

2. Predict-and-Correct-m-Times, PC^m. This procedure is employed when a variable-step-size technique is used. Setting m = 3 allows the corrections to be carried out up to three times.

3. Self-Starting Procedure. First order generalized Adams-Bashforth method is used as a self-starter. The step size used for the starter is 10 times smaller than the user-suggested step size.

4. A is a Function of t.[11] In solving stiff equations of the form

$$y' = A(t)y + g(t,y) , \tag{4-1}$$

the program is constructed to decompose the equation into

$$y' = A(t_i)y + \{A(t) - A(t_i)\}y + g(t,y) , \tag{4-2}$$

so that the NLMS methods can be applied. Since the above equation is stiff, $Re\{\lambda(A(t))\} < 0$ for all t. If $||A(t) - A(t_i)||$ can be maintained small enough, the above problem can be solved accurately. The decomposition is performed automatically, provided the indicator IAT is set to 1. As one can see, after the decomposition, the new g(t,y) becomes $\{A(t) - A(t_i)\}y + g(t,y)$; only the original g(t,y) needs to be defined in the GFN subroutine. The HMAX is recommended such that $||A(t) - A(t_i)|| < hP$. An example is given in the section of numerical examples.

INPUT INFORMATION

A listing of inputs is given below with their definitions:

ERR	User-required tolerance
T(1)	Initial time
TMAX	Final time
IPC	= 0, fixed-step size
	≠ 0, variable-step size
INDEX	= 0, explicit methods
	≠ 0, implicit methods
KSTEP	Step number (also order of the method < 4)

H Initial step size

HMAX Largest step size allowed

HMIN Smallest step size allowed

N Order of the system

YZERO A vector array to store the initial values.

Some default values are built in, in case the user failed to supply them. They are

HMAX = 0.1 D-0

HMIN = 0.1 D-5

ERR = 0.1 D-7

H = 0.1 D-1.

5. AN APPLICATION - AN UNDERWATER WAVE PROPAGATION EXAMPLE

The underwater acoustic wave propagation problems can be expressed mathematically by an elliptic equation,

$$\Delta^2 p + k^2 n^2 p = 0 \ , \tag{5-1}$$

where

p is the acoustic pressure,

k is the wavenumber, and

n = n(r,z) is the index of refraction.

Assuming the inhomogeneities are slowly varying in range, a cylindrically symmetric geometry, and a harmonic source, equation (5-1) can be transformed into two uncoupled parabolic wave equations. One parabolic equation stands for the transmitted field, representing the wave propagating in the range direction. The other parabolic equation stands for the reflected field, representing the wave propagating in the depth direction. Because of the slowly varying property of inhomogeneities in range, the reflected field is negligible. Therefore, one needs only to solve the transmitted field, and Tappert[13] did this.

Tappert's algorithm is known as the split-step Fourier algorithm and is considered to be the only effective technique for solving the parabolic wave equations in the underwater acoustic field, with some limitations. For one class of problems, GAB-GAM methods can produce reasonably accurate solutions rapidly without restrictions.

First, let us take the ODE approach to treat the solution of parabolic wave equations. We obtain a system of ODE's where the second partial derivative with respect to the space variable is discretized by means of a second order central difference, as follows.

A parabolic wave equation in one-space dimension takes the form

$$u_r = a(k_0,r,z)u + b(k_0,r,z)u_{zz} \ . \tag{5-2}$$

Discretizing u_{zz} by a second order central difference brings equation (5-2) to

$$\frac{du_m}{dr} = a_m u_m + \frac{b_m}{H^2}[u_{m+1} - 2u_m + u_{m-1}] \ . \tag{5-3}$$

Equation (5-3) is a system of first order ODE's that can be put in the matrix form

$$\begin{bmatrix} \frac{du_1}{dr} \\ \frac{du_2}{dr} \\ \vdots \\ \frac{du_{m-1}}{dr} \\ \frac{du_m}{dr} \end{bmatrix} = \begin{bmatrix} a_1 - \frac{2b_1}{H^2} & \frac{b_1}{H^2} & 0 & \cdots & 0 & 0 \\ \frac{b_2}{H^2} & a_2 - \frac{2b_2}{H^2} & \frac{b_2}{H^2} & \cdots & 0 & 0 \\ & & & \vdots & & \\ 0 & 0 & 0 & \cdots & a_{m-1} - \frac{2b_{m-1}}{H^2} & \frac{b_{m-1}}{H^2} \\ 0 & 0 & 0 & \cdots & \frac{b_m}{H^2} & a_m - \frac{2b_m}{H^2} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{m-1} \\ u_m \end{bmatrix} + \begin{bmatrix} \frac{b_1}{H^2}u_0 \\ 0 \\ \vdots \\ 0 \\ \frac{b_m}{H^2}u_{m+1} \end{bmatrix} , \quad (5\text{-}4)$$

where u_0, u_{m+1} are two boundary points at range r. To put the system above in a short matrix form, we get

$$u' = A(r,z)u + g(r,z,u) . \quad (5\text{-}5)$$

In selecting a range step size for solving equation (5-4), using GAB-GAM methods with a constant matrix A, one will choose

$$h\left|\left|\frac{\partial g}{\partial u}\right|\right|\left|\left|\phi_{kk}(Ah)\right|\right| < 1 \quad (5\text{-}6)$$

for corrector's convergence.

In the range-independent case, g can be decomposed again such that $||\partial g/\partial u|| = 0$. This implies that a large step size can be used.[11] Note, also, that the ODE approach formulates the problem without other restrictions.

A first order GAB-GAM is used with the predictor-and-corrector feature to solve the following problem. The equation

$$u_1 = \frac{ik_0}{2}(n^2(r,z) - 1)u + \frac{i}{2k_0}u_{zz} \quad (5\text{-}7)$$

describes a shallow-water wave propagation in the range direction in the region indicated in figure 1.

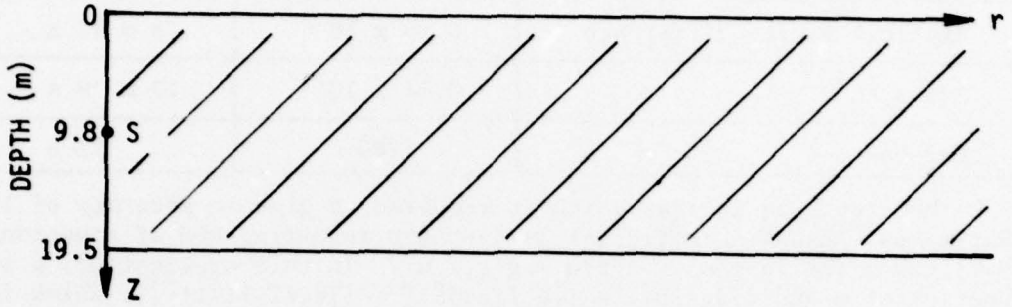


Figure 1. Region of Wave Propagation

The source is placed at 9.8 m below the surface and the bottom depth is 19.5 m. The bottom is rigid; thus, a Neumann boundary condition is assumed. We examine the wave propagation up to 400 m. The sound speed profile is described in figure 2.

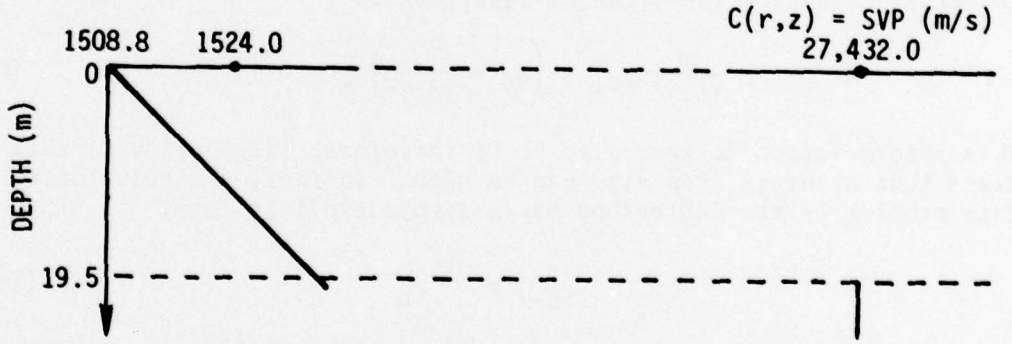


Figure 2. Sound Speed Profile

The initial sound speed C_0 = 1.5 km/s, frequency = 25 Hz, the surface condition is u = 0, and $n(r,z) = c(r,z)/c_0$.

The solution was obtained through four different methods: normal mode,[14] explicit finite-difference, first order Adams PC^3, and the first order GAB-GAM. The normal mode solution is very suitable for this problem; therefore, we use it as our reference solution for comparison of accuracy. Solutions by explicit finite-difference, Adams PC^3, and GAB-GAM all agree almost exactly with the normal mode solution; the difference is in the speed, as tabulated below:

	h	Time
Explicit finite-difference	0.35×10^{-5}	8 m 11 s
Adams PC[3]	0.34×10^{-4}	10 m 9 s
GAB-GAM	1280	23 s

The step size is that which is required to give an accuracy of 10^{-4}. What makes GAB-GAM so efficient is that the resulting ODE of equation (5-7) takes the form $u' = A(z)u + g(r,z,u)$. In this application, g is a function of u and $||\partial g/\partial u|| = ||b_m/(\Delta z)^2|| = ||c_0/2\pi f(\Delta z)^2||$, which is not a small number for reasonably small (Δz). However, the rigid bottom allows us to restate the problem in a desirable way. Notice that the last equation of the system is

$$\frac{b_m}{(\Delta z)^2}u_{m-1} + \left(a_m - \frac{2b_m}{(\Delta z)^2}\right)u_m + \frac{b_m}{(\Delta z)^2}u_{m+1} = 0 \ . \qquad (5\text{-}8)$$

The rigid bottom condition enables u_{m+1} to be expressed as u_m; therefore, equation (5-8) can be rewritten as

$$\frac{b_m}{(\Delta z)^2}u_{m-1} + \left(a_m - \frac{b_m}{(\Delta z)^2}\right)u_m = 0 \ . \qquad (5\text{-}9)$$

This reformulation defines g to be 0; therefore, $||\partial g/\partial u|| = 0$; this means that a larger step size can be used. In fact, the solution to this problem by the GAB method has a simple explicit form,

$$y_{n+1} = e^{Ah}y_n \ . \qquad (5\text{-}10)$$

Hence, subject to the accurate computation of e^{Ah}, the solution can be found in one step. We can see that for a range-independent matrix A and rigid plane parallel boundaries, GAB-GAM methods have definite advantages.

6. NUMERICAL EXAMPLES

GAB-GAM methods are designed to solve stiff ODE's in the form $y' = Ay + p(t)$, where $p(t)$ is a low order polynomial in t and A is a constant matrix, in the absence of roundoff and Pade approximation errors. In the following examples, we present a semistiff system, a nonlinear stiff system, and a stiff system with $A = A(t)$. The last two examples are not in the format recognizable by NLMS methods. We will show how these problems can be brought into equivalent forms such that GAB-GAM is applicable. The starter employed is a first order GAB with a step size 10 times smaller than the initial step size user suggested, and all the examples are started at time = 0. For every fixed-step size, only one matrix inversion is needed for the algorithm. An additional matrix inversion is needed for the Pade approximation. Therefore, only two matrix inversions are needed to solve the whole problem for a fixed-step size.

EXAMPLE 1, SEMISTIFF SYSTEM[10]

Problem

$$y' = \begin{bmatrix} 0 & 1 \\ 10 & -9 \end{bmatrix} y + \begin{bmatrix} 1 \\ 1 \end{bmatrix}; \quad y(0) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

Exact Solution

$$y(t) = \{2e^t - 1, 2e^t - 1\}^T .$$

Eigenvalues of A

$$\{1, -10\} .$$

Results

Method (Order)	t_{max}	Step Size	Number of Steps	Number of g Calls	Number of Inversions	Error	
GAB(3)	10	10	1	3	2	41×10^{-12}	GAB
						0	Exact

Remarks

NLMS methods are designed to solve this type of problem exactly, subject to the accurate computation of e^{Ah}. In this case, it is expected that NLMS methods will produce accurate results with large step size without requiring large computational expense.

EXAMPLE 2, NONLINEAR STIFF SYSTEM[15,16,17]

Problem

$$y_1' = -0.013y_2 - 1000y_1y_2 - 2500y_1y_3 \; ; \; y_1(0) = 0 \; ,$$

$$y_2' = -0.013y_2 - 1000y_1y_2 \; ; \; y_2(0) = 1 \; ,$$

$$y_3' = -2500y_1y_3 \; ; \; y_3(0) = 1 \; .$$

Reference Solution

$$y_1(48) = -0.194533895680D\text{-}5 \; ,$$

$$y_2(48) = 0.611474831446 \; ,$$

$$y_3(48) = 1.388950571516 \; .$$

The above problem is in the form $y' = f(t,y)$; $y(t_0) = y_0$. This problem is not in the form directly acceptable by NLMS methods. Since the system is autonomous, we can easily restate it appropriately. Transform the dependent variable such that $z_1(t) = y_1(t)$; $z_2(t) = y_2(t) - 1$; $z_3(t) = y_3(t) - 1$. Consequently, $z(0) = 0$. We now expand $f(z)$ about the critical point; i.e.,

$$z' = f(z) = f(0) + zf'(0) + 0(z^2) = f'(0)z + g(z) \; ,$$

where $A = f'(0) = (\partial f_i/\partial z_j)_{z=0}$ and $g(z) = f(0) + 0(z^2)$. The matrix thus found is singular. The value of a_{33} is adjusted to -0.01 to keep A nonsingular while similarly readjusting $g(z)$. Therefore, we obtain an equivalent system, as follows,

$$\begin{bmatrix} z_1' \\ z_2' \\ z_3' \end{bmatrix} = \begin{bmatrix} -3500 & -0.013 & 0 \\ -1000 & -0.013 & 0 \\ -2500 & 0 & -0.01 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} + \begin{bmatrix} -0.013 - 1000z_1z_2 - 2500z_1z_3 \\ -0.013 - 1000z_1z_2 \\ -2500z_1z_3 + 0.01z_3 \end{bmatrix},$$

with $z_1(0) = z_2(0) = z_3(0) = 0$.

Eigenvalues of A

$$\{-0.00993,\ -0.01,\ -3500.01307\}\ .$$

Results

Method (Order)	t_{max}	Step Size	Number of Steps	Number of g Calls	Number of Matrix Inversions	Error	
GAB(2)	48	16	2	4	2	89×10^{-4}	GAB(2)
						70×10^{-4}	Jeltsch

Remarks

For autonomous systems, an expansion about the critical point of the right-hand-side function produces the desired format for GAB-GAM methods.

EXAMPLE 3, STIFF SYSTEM WITH A = A(t)[5,8,18]

Problem

$$y' = \begin{bmatrix} -60 + 0.125t & 10 \\ 0.2 & -0.2 \end{bmatrix} y + \begin{bmatrix} 0.125t \\ 0 \end{bmatrix};\ y(0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

Reference Solution

$$y_1(10) = 2.344886 \times 10^{-2},$$

$$y_2(10) = 1.301528 \times 10^{-2},$$

$$y_1(400) = 27.110701,$$

$$y_2(400) = 22.242211\ .$$

Introducing a new variable, $y_3 = t$, brings the above problem into an autonomous system,

$$y' = \begin{bmatrix} -60 & 10 & 0.125 \\ 0.2 & -0.2 & 0 \\ 0 & 0 & -1 \end{bmatrix} y + \begin{bmatrix} 0.125 y_1 y_3 \\ 0 \\ 1 + y_3 \end{bmatrix} ; \; y(0) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} .$$

Eigenvalues of A

$$\{-0.2335, -1, -59.9665\} .$$

Results

Method (Order)	t_{max}	Step Size	Number of Steps	Number of g Calls	Number of Matrix Inversions	Error	
GAM(3)	10	1	7	53	2	68×10^{-11}	GAM(3)
						49×10^{-9}	Lambert
	400	1	397	2783	2	50×10^{-3}	GAM(3)
						44×10^{-4}	Lambert

Remarks

NLMS methods are developed for constant A. For A = A(t), the problem is restated in a form acceptable to NLMS. The main strategy is to construct g(t,y) to be a low order polynomial in t or a slowly varying function in t. Note that the solution at t = 10 is almost two orders of magnitude better than Lambert's, despite the large step size. The comparison at t = 400 given by Lambert may itself be in error since he used Runge-Kutta methods to achieve y(400).

7. OTHER ADAMS-LIKE DEVELOPMENTS

CERTAINE'S DEVELOPMENT[2]

Certaine considered the differential equation in the form

$$y' = -Dy + g(t,y) \ , \tag{7-1}$$

where D is a diagonal matrix. Multiplying both sides of equation (7-1) by e^{Dt} and integrating it over the interval $[t_1,t_2]$, Certaine obtained

$$y_2 = e^{-Dh}y_1 + \int_{t_1}^{t_2} e^{D(t'-t_2)}g(t',y(t'))dt' \ , \tag{7-2}$$

where $h = t_2 - t_1$; $g(t',y(t'))$ is approximated by a polynomial in t', call it a(t'), of degree n + 1; and n is restricted to be ≤ 2.

The integral of equation (7-2) can be written as

$$\int_{t_1}^{t_2} e^{D(t'-t_2)}g(t',y(t'))dt' \approx \sum_{j=0}^{n} C_j a_j \ , \tag{7-3}$$

where

$$g(t,y(t)) \approx a(t) = \sum_{j=0}^{n} \frac{(t - t_1)^j}{j!h^j} a_j \ ,$$

and

$$C_j = \int_{t_1}^{t_2} e^{D(t'-t_2)} \frac{(t' - t_2)^j}{j!h^j} dt' \ .$$

It is immediately seen that

$$C_0 = D^{-1}(I - e^{-Dh})$$

and

$$C_{n+1} = D^{-1}\left(\frac{I}{(n+1)!} - h^{-1}C_n\right) .$$

For the first order method,

$$y_2 = e^{-Dh}y_1 + (C_0 - C_1)g_1 + C_1g_2 . \tag{7-4}$$

For the second order method,

$$y_2 = e^{-Dh}y_1 + \left(C_2 - \tfrac{1}{2}C_1\right)g_0 + \left(C_0 - 2C_2\right)g_1 + \left(\tfrac{1}{2}C_1 + C_2\right)g_2 . \tag{7-5}$$

If we take A = -D, simplify $(C_0 - C_1)$, and substitute it into equation (7-4), we get

$$y_2 = e^{Ah}y_1 + h\{-(Ah)^{-2}[[(I - Ah)e^{Ah} - I]g_1 + [(I + Ah) - e^{Ah}]g_2]\}, \tag{7-6}$$

which is exactly the first order GAM method. Further, it can be verified that Certaine's method of order 2 is the second order GAM method. Therefore, we see that the GAB-GAM methods include Certaine's work.

BJUREL'S MODIFICATION[3]

Bjurel considered the differential equation of the form

$$P_N(d/dt)y = f(t,y) , \tag{7-7}$$

where P is a polynomial of degree n with constant coefficients. Bjurel generalized LMS methods and obtained

$$\sum_{i=0}^{k}[\alpha_i(h)y_{n+i} - \beta_i(h)f_{n+i}] = 0;\ \alpha_k(h) = 1 , \tag{7-8}$$

where $\alpha_i(h)$, $\beta_i(h)$ are mesh dependent.

Bjurel defined

$$\rho(\zeta,h) = \sum_{i=0}^{k}\alpha_i(h)\zeta^i ,$$

$$\sigma(\zeta,h) = \sum_{i=0}^{k} \beta_i(h)\zeta^i \ .$$

In order to give rise to the Adams-like methods, Bjurel first considered $\rho(\zeta,h)$ is of the form

$$\rho(\zeta,h) = \zeta^k \prod_{j=1}^{N} (1 - e^{\lambda_j h}\zeta^{-1}) \ , \tag{7-9}$$

where λ_j are roots of $P_N = 0$, and then selected the remaining $(k - n)$ roots of $P_N = 0$.

The resemblance to the ordinary Adams formulas can be seen by putting $N = 1$ and $\lambda_1 = 0$.

NORSETT'S METHOD[4]

Norsett's A-stable modification of the Adams-Bashforth methods considered the problem

$$y' = f(t,y); \ y(t_0) = y_0 \tag{7-10}$$

over the interval $I = [t_0,b]$.

Choose $t_n \epsilon I$ and define $t_{n+i} - t_n = ih$. Write $y' = f(t,y)$ as

$$y' + Py = Py + f(t,y) \ , \tag{7-11}$$

where P is a constant matrix. Approximate $T(y) = y' + Py$ with a Lagrange interpolation polynomial at points t_{n+i}, $i = 0,1,\ldots,q$.

Integrating $T(t)$ between t_n and t_{n+i} and using methods of generating functions, we obtain the following formula:

$$y_{n+1} = e^{-Ph}y_n + h \sum_{i=0}^{q} \beta_{qi}^{s}(f_{n-i} + Py_{n-i}) \ , \tag{7-12}$$

where

$$\beta_{qi}^{s} = (-1)\sum_{m=i}^{q} s_m \binom{m}{i}, \quad s_m = (-1)^m \int_0^1 e^{Phs} \binom{-s}{m} ds, \quad s = \frac{t - t_n}{h},$$

$q = 0,1,2,\ldots;\ i = 0,1,\ldots,q.$

It is easily seen that if we take $q = 0$, $i = 0$, $g_{n-1} = f_{n-1} + Py_{n-1}$, and $P = A$, we get

$$y_{n+1} = e^{-Ph} y_n + h(Ah)^{-1}[I - e^{Ah}] f_n . \tag{7-13}$$

This is identical to the first order GAB methods.

JAIN'S METHODS[6]

Using a similar starting point as Norsett, Jain developed A-stable methods for stiff ODE's based on Hermite interpolation polynomials. Let

$$\ell_j(t) = \frac{(t - a_1)(t - a_2)\cdots(t - a_{j-1})(t - a_{j+1})\cdots(t - a_n)}{(a_j - a_1)(a_j - a_2)\cdots(a_j - a_{j-1})(a_j - a_{j+1})\cdots(a_j - a_n)} \tag{7-14}$$

be the Lagrange interpolation polynomials.

Let

$$h_j(t) = [1 - (t - a_j)\ell_j^1(a_j)]\ell_j^2(t) \tag{7-15}$$

and

$$\bar{h}_j(t) = (t - a_j)\ell_j^2(t) \tag{7-16}$$

be Hermite interpolation formulas. Write $y' = f(t,y)$ as

$$y' + Py = Py + f(t,y) . \tag{7-17}$$

Approximating $y' + Py$ by Hermite interpolation polynomials yields

$$y'(t) + Py(t) = \sum_{i=1}^{n} h_i(t)(f_i + Py_i) + \sum_{i=1}^{n} \bar{h}_i(t)(f_i' + Pf_i) + R_n , \quad (7\text{-}18)$$

where

$$R_n = \frac{1}{(2n)!} F^{(2r)}(\xi)\pi^2(t) ,$$

$$F(t) = f(t) + Py(t) ,$$

and

$$\pi(t) = (t - a_1)(t - a_2) \cdots (t - a_n) . \quad (7\text{-}19)$$

Integrating equation (7-18) from t_n to t_{n+1} gives

$$y_{n+1} = e^{-Ph} y_n + e^{-Pt_{n+1}} \left[\sum_{i=1}^{n} (H_i F_i + \bar{H}_i F_i') \right] + R_n , \quad (7\text{-}20)$$

where

$$H_i = \int_{t_n}^{t_{n+1}} e^{Pt} h_i(t)dt , \quad (7\text{-}21)$$

$$\bar{H}_i = \int_{t_n}^{t_{n+1}} e^{Pt} \bar{h}_i(t)dt , \quad (7\text{-}22)$$

$$R_n = \frac{e^{-Pt_{n+1}}}{(2n)!} \int_{t_n}^{t_{n+1}} e^{Pt} F^{(2n)}(\xi)\pi^2(t)dt , \quad (7\text{-}23)$$

P is an approximation to $-(\partial f/\partial y)_n$ (which is chosen to be

$$P = - \frac{f_n - f(t_n, y_{n-1})}{y_n - y_{n-1}} \quad (7\text{-}24)$$

for a scalar equation),

$$P_{ii} = - \frac{f_n^i - f^i(t_n, y_n^1, \cdots y_{n-1}^i, \cdots y_n^s)}{y_n^i - y_{n-1}^i} \qquad (7\text{-}25)$$

for a system of equations, and P is considered to be a diagonal matrix.

Equation (7-20) can be rewritten, by changing variables and letting $s = (t - t_n)/h$, as

$$y_{n+i} = e^{-Ph} y_n + h e^{-Ph} \left[\sum_{i=1}^{n} (H_i F_i + \overline{H}_i F_i') \right] + R_n , \qquad (7\text{-}26)$$

where

$$H_i = \int_0^1 e^{Phs} h_i(s) ds , \qquad (7\text{-}27)$$

$$\overline{H}_i = \int_0^1 e^{Phs} \overline{h}_i(s) ds , \qquad (7\text{-}28)$$

$$R_n = h^{2n+1} e^{-Ph} F^{(2n)}(\xi) \left\{ \frac{1}{(2n)!} \int_0^1 \pi^2(s) ds \right\} + 0(h^{2n+2}) . \qquad (7\text{-}29)$$

Therefore, the method (7-20) is of order 2n.

The integrations involved to determine $H_i, \overline{H}_i$ are of the form

$$I_n = \int_0^1 e^{Phs} \left(\sum_{i=1}^{N} C_i s^i \right) ds , \qquad (7\text{-}30)$$

N = 0,1,...,n.

Take a simple case, n = 1, and we have $h_1(t) = 1$; $\bar{h}_1(t) = t - t_1$; $\pi(t) = t - t_1$; $h_1(s) = 1$; $\bar{h}_1(s) = s$; and $\pi(s) = s$. Then

$$H_1 = \int_0^1 e^{Phs} ds = \frac{1}{Ph}(e^{Ph} - 1) ,$$

$$\bar{H}_1 = \int_0^1 s e^{Phs} ds = \left(\frac{1}{Ph} - \frac{1}{P^2h^2}\right)e^{Ph} + \frac{1}{P^2h^2} ,$$

and

$$\frac{1}{(2n)!}\int_0^1 \pi^2(s) ds = \frac{1}{6} .$$

Select the coefficients of $\bar{H}_1$ to be 0, and equation (7-26) becomes

$$y_{n+1} = e^{-Ph} y_n + he^{-Ph}\left\{(Ph)^{-1}(e^{Ph} - 1)\right\}F_n . \qquad (7\text{-}31)$$

Define -P = A and $F_n = g_n$, and equation (7-31) is identical to the first order GAB methods.

MURPHY'S DEVELOPMENT[7]

Murphy proceeded again from the same starting point as Norsett and Jain. Murphy wrote y' = f(t,y) as

$$y' + Ly = Ly + f(t,y) . \qquad (7\text{-}32)$$

Multiplying both sides of equation (7-32) by e^{Lt} and integrating from t_n to t_{n+1} yields

$$y(t_{n+1}) = e^{-Lh} y(t_n) + e^{-t_{n+1}} \int_{t_n}^{t_{n+1}} e^{Lt} f(t,y) dt . \qquad (7\text{-}33)$$

Murphy employed Newton's divided difference representation of the Hermite interpolation polynomial of degree 2r to treat the integral.

A change of variable, letting $s = (t - t_n)/h$ in the integral of equation (7-33), gives

$$e^{Lt_{n+1}} \int_{t_n}^{t_{n+1}} e^{Lt} f(t,y)dt = he^{-Lh} \int_{t_n}^{t_{n+1}} e^{Lsh} f(t_n + sh, y(t_n + sh))dt. \quad (7\text{-}34)$$

The function $f(t_n + sh, y(t_n + sh))$ is approximated by Newton's divided difference formula, using the values $t_n, t_n, t_{n-1}, t_{n-1}, \cdots, t_{n-r+1}, t_{n-r+1}$. Therefore,

$$\begin{aligned} f(t_n + sh, y(t_n + sh)) &= f_n + shf_{n,n} + s^2h^2 f_{n,n,n-1} \\ &\quad + (s+1)s^2h^3 f_{n,n,n-1,n-1} + \cdots \\ &\quad + (s+1)^2 s^2 h^{2r-1} f_{n,n,\cdots,n-r+1,n-r+1} + R_n \,. \end{aligned} \quad (7\text{-}35)$$

The error term

$$R_n = [s(s+1)\cdots(s+r-1]^2 h^{2n} f[t_n, t_n, \cdots, f_{n-r+1,n-r+1}] \,.$$

The $f_{i,i+1,\cdots,i+j}$ in equation (7-35) is the j-th Newton's divided difference given by the quotient

$$f_{i,i+1,\cdots,i+j} = \frac{f_{i+1,\cdots,i+j} - f_{i,\cdots,i+j-1}}{t_{i+j} - t_i} \,,$$

and $f_{n,n}$ is defined to be equal to $f'(t_n)$.

The Murphy method takes the form

$$\begin{aligned} y_{n+1} &= e^{-Lh} y_n + h[a_0 f_n + a_1 f_{n,n} + a_2 f_{n,n,n-1} + \cdots \\ &\quad + a_{2r-1} f_{n,n,n-1,n-1,\cdots,n-r+1,n-r+1}] \,, \end{aligned} \quad (7\text{-}36)$$

where

$$a_i = h^i e^{-Lh} \int_0^1 P_i(s) e^{Lsh} ds \,,$$

$$= \frac{h^i}{Lh}\left[\sum_{j=0}^{i}(-1)^j \frac{p_i^{(j)}(1)}{(Lh)^j} - \sum_{j=0}^{i}(-1)^j \frac{p_i^{(j)}(0)e^{-Lh}}{(Lh)^j}\right], \tag{7-37}$$

and the $P_m^{(n)}(s)$ can be calculated by the recursive formulas below:

$$P_{2m-1}^{(n)}(s) = (s + m - 1)P_{2m-2}^{(n)}(s) + nP_{2m-2}^{(n-1)}(s) , \tag{7-38}$$

$m = 1,2,\cdots,r$ $n = 0,1,2,\cdots,2m-1$;

$$P_{2m}^{(n)}(s) = (s + m - 1)P_{2m-1}^{(n)}(s) + nP_{2m-1}^{(n-1)}(s) , \tag{7-39}$$

$m = 1,2,\cdots,r$, $n = 0,1,2,\cdots,2m$; with $P_0^{(0)}(s) = P_0(s) = 1$, $P_m^{(-1)}(s) = 0$.

The choice of $a_j = 0$ for $j \geq 1$ produces equation (7-40) from equation (7-36), which coincides with the first order GAB methods for $j = 0$. This choice gives

$$\begin{aligned} a_0 &= (Lh)^{-1}(1 - e^{-Lh}) , \\ y_{n+1} &= e^{-Lh}y_n + h\left\{(Lh)^{-1}(1 - e^{-Lh})\right\}f_n . \end{aligned} \tag{7-40}$$

VERWER'S FORMULATION[5]

Consider an LMS formula of the type

$$y_{n+1} = \sum_{\ell=1}^{k} A_\ell(h_nJ_n)y_{n+\ell-1} + h_n\sum_{\ell=1}^{k} B_\ell(h_nJ_n)f_{n+1-\ell} , \tag{7-41}$$

where J_n is a matrix that equals or approximates $J(t_n,y_n)$, the Jacobian matrix $\partial f/\partial y$. Based on approximate choice of rational function, Verwer arrived at a multistep formula with zero-parasitic roots and an adaptive principal root from equation (7-41). This formula leads to

$$y_{n+1} = R(h_nJ_n)y_n + h_n\sum_{\ell=1}^{k} B_\ell(h_nJ_n)[f_{n+1-\ell} - J_ny_{n+1-\ell}] \tag{7-42}$$

Equation (7-41) can be put in the form

$$y_{n+1} = Py_{n+1-\ell} + h_n \sum_{\ell=1}^{k} B_\ell(h_n J_n) g_{n+1-\ell} , \qquad (7\text{-}43)$$

where

$$P = \sum_{\ell=1}^{k} \left\{ A_\ell(h_n J_n) + h_n B_\ell(h_n J_n) M_{n+1-\ell} \right\} ;$$

$$M_{n+1-\ell} y_{n+1-\ell} + f_{n+1-\ell} = g_{n+1-\ell} .$$

If we use a Pade approximation for P and let $B_\ell(h_n J_n)$ be $\phi_{k\ell}(Mh)$, equation (7-43) is identical to the GAB method. Note that Verwer does not consider implicit techniques at all.

LAMBERT'S METHODS[8]

To solve the problem $y' = f(t,y)$; $y(t_0) = y_0$, consider a class of multistep methods with variable matrix coefficients, as defined below:

$$\sum_{i=0}^{k} [\alpha_i + ha_i(t_n)] y_{n+i} = h \sum_{i=0}^{k} [\beta_i + hb_i(t_n)] f_{n+i} , \qquad (7\text{-}44)$$

where $y_i = y(t_i)$; $f_i = f(t_i, y_i)$; $t_i = t_0 + ih$; $|a_i(t)| \leq$ a constant; and $|b_i(t)| \leq$ a constant for every $t \varepsilon [t_0, t_{max}]$.

A class of the above methods has been considered, as follows:

$$\sum_{i=0}^{k} \left[\alpha_i^{(0)} I + \sum_{s=1}^{s} a_i^{(s)} h^s Q_n^s \right] y_{n+i} = h \sum_{i=0}^{k} \left[b_i^{(0)} I + \sum_{s=1}^{s} b_i^{(s)} h^s Q_n^s \right] f_{n+i} , \qquad (7\text{-}45)$$

and $||Q_n|| \leq q$ and $||a_k^{(0)} I + \sum_{s=1}^{s} a_k^{(s)} h^s Q_n^s||$ are nonsingular.

To label a class of the above methods as Method 1, Lambert selected $s = 1$; $k = 1$; $p = 1,2$; $a_i^{(0)} = 1$; $a_0^{(0)} = -1$; $b_1^{(0)} = 1/2$; $b_0^{(0)} = 1/2$;

$a_1^{(1)} = 0$; and $a_0^{(1)} = 0$. This selection gives the first order Adams-Moulton method. A class of these methods is used to solve problems in examples 2 and 3 of section 6 with

$$Q_n = - \left. \frac{\partial f}{\partial y} \right|_{t = t_{n+2}}$$

8. CONCLUSIONS

NLMS methods have the advantage of allowing the use of a large step size for solving stiff equations; GAB-GAM methods have the advantage that they are strongly stable and require the least number of operations in the class of NLMS methods.

A good implementation of GAB-GAM methods, certainly, will solve initial value problems of stiff equations effectively. GAB-GAM methods are not restricted to solving stiff equations only, as example 1 of section 6 shows. However, the GAB-GAM methods are not generally competitive with ordinary Adams methods for such problems.

Pade approximation plays an important role in the approximation of e^{Ah}. A number of formulas[19] have been given that approximate e^{Ah} efficiently. The PADE subroutine listed in this report is for eigenvalues being negatively large and differing greatly in magnitude. For mild or semistiff problems, other Pade formulas may be substituted.

GAB-GAM methods are designed to handle ODE's in the form $y' = Ay + g(t,y)$, where A is a constant and nonsingular matrix. For $A = A(t)$, a decomposition can be made to transform the problem into a form acceptable to GAB-GAM methods.

REFERENCES

1. F. T. Krogh, "On Testing a Subroutine for the Numerical Integration of Ordinary Differential Equations," J. of the ACM, vol. 20, no. 4, 1973, pp. 543-562.

2. J. Certaine, "The Solution of Ordinary Differential Equations With Large Time Constants," in Mathematical Methods for Digital Computers, ed. A. Ralston and H. Wilf, John Wiley and Sons, Inc., NY, 1960, pp. 128-132.

3. G. Bjurel, Modified Multistep Methods for a Class of Stiff Ordinary Differential Equations, Report NA 71.42, Dept. of Info. Proc., The Royal Institute of Technology, Stockholm, Sweden, 1971.

4. S. P. Norsett, "An A-Stable Modification of the Adams-Bashforth Methods," Conf. on the Numer. Sol. of Diff. Eqs., 109, Spring-Verlag, New York, 1969.

5. J. G. Verwer, "On Generalized Linear Multistep Methods with Zero-Parasitic Roots and an Adaptive Principal Root," Numerische Mathematik, vol. 27, Fasc 2, 1977, pp. 143-155.

6. R. K. Jain, "Some A-Stable Methods for Stiff Ordinary Differential Equations," Math. Comp., vol. 26, no. 117, 1972, pp. 71-77.

7. W. D. Murphy, "Hermite Interpolation and A-Stable Methods for Stiff Ordinary Differential Equations," Appl. Math. & Comp., 3, 1977, pp. 103-112.

8. J. D. Lambert and S. T. Sigurdson, "Multistep Methods With Variable Matrix Coefficients," SIAM J. Numer. Anal. 9, 1972, pp. 715-733.

9. P. Henrici, Discrete Variable Methods in Ordinary Differential Equations, John Wiley and Sons, Inc., NY, 1962.

10. D. Lee, "Nonlinear Multistep Methods for Solving Initial Value Problems in Ordinary Differential Equations," Ph.D. Dissertation, Polytechnic Institute of New York, 1974.

11. D. Lee and S. Preiser, "A Class of Strongly Stable and A-Stable Numerical Methods for Solving Stiff Equations," J. of Comp. & Math. with Appl., vol. 4, no. 1, 1978, pp. 43-51.

12. G. Dahlquist, "A Special Stability Problem for Linear Multistep Methods," BIT 3, 1963, pp. 27-43.

13. R. H. Hardin and F. D. Tappert, "Applications of the Split-Step Fourier Method to the Numerical Solution of Nonlinear and Variable Coefficient Wave Equations," SIAM Rev. 15, 1973, p. 423.

14. W. G. Kanabis, A Shallow Water Acoustic Model for an Ocean Stratified in Range and Depth, NUSC Technical Report 4887-I, Naval Underwater Systems Center, New London, CT, 1975.

15. H. Brunner, "Recursive Collocation for the Numerical Solutions of Stiff Ordinary Differential Equations," Math. Comp., vol. 28, no. 126, 1974, pp. 475-481.

16. C. W. Gear, "The Automatic Integration of Stiff Ordinary Differential Equations," Info. Proc. 68, North Holland Publishing Co., Amsterdam, The Netherlands, 1969, pp. 187-193.

17. R. Jeltsch, "Multistep Methods Using Higher Derivatives and Damping at Infinity," Math. Comp., vol. 26, no. 117, 1977, pp. 71-77.

18. W. Liniger and R. Willoughby, Efficient Numerical Integration of Stiff Systems of Ordinary Differential Equations, Research Report RC 1970, IBM, Yorktown Heights, NY, 1967.

19. J. L. Blue and H. K. Gummel, "Rational Approximations to Matrix Exponential for Systems of Stiff Equations," J. of Comp. Physics, 5, 1970, pp. 70-83.

Appendix

A LISTING OF COMPUTER PROGRAMS

A listing of ANSI FORTRAN computer programs follows. Among the programs, the matrix inversion subroutine is not listed. The user can supply a matrix inversion from his computer library. The main program describes the solution setup of example 2, from section 6 of the main text. This problem can be used as a test case.

```
C
C     ***********************************************************
C     *   THIS PROGRAM COMMANDS A SET OF SUBROUTINES TO INTEGRATE*
C     *   A SYSTEM OF FIRST-ORDER ODE OF THE FORM               *
C     *                                                         *
C     *                     DY(T)/DT=AY + G(T,Y)                *
C     *                                                         *
C     *   OVER THE CLOSED INTERVAL (T(1),TMAX).                 *
C     *   OPTIONS ARE : VARIABLE-STEP-SIZE (FIXED STEP SIZE AS WELL),*
C     *                 SELF-STARTING, PREDICTOR-CORRECTOR,     *
C     *                 MATRIX A IS A FUNCTION OF T             *
C     *   INPUT PARAMETERS HAVE THE FOLLOWING DEFINITIONS:      *
C     *                                                         *
C     *   A       A 2-DIMENSIONAL ARRAY MATRIX                  *
C     *                                                         *
C     *   ERR     USER-SPPLIED ERROR TOLERANCE                  *
C     *           DEFAULT VALUE = 0.1D-8                        *
C     *                                                         *
C     *   H       INITIAL STEP SIZE, DEFAULT VALUE=0.001        *
C     *                                                         *
C     *   HMAX    THE MAXIMUM STEP SIZE THE USER ALLOWS THE PROGRAM*
C     *           TO CONSIDER. DEFAULT VALUE=1.D0               *
C     *                                                         *
C     *   IAT     SET=0, IF A IS A CONSTANT                     *
C     *           SET=1, IF A IS A FUNCTION OF T                *
C     *                                                         *
C     *   IGFN    SET=0, IF G(T,Y)=0                            *
C     *           SET=1, IF G(T,Y) NOT = 0                      *
C     *                                                         *
C     *   INDEX   SET=0, ASKING FOR EXPLICIT METHODS            *
C     *           SET=1, ASKING FOR IMPLICIT METHODS            *
C     *                                                         *
C     *   IPC     SET=0 FOR PRIDICTOR OR  CORRECTOR             *
C     *           SET=1 FOR PREDICTOR-AND-CORRECTOR             *
C     *                                                         *
C     *   KSTEP   STEP NUMBER (LESS THAN 4)                     *
C     *                                                         *
C     *   N       NUMBER OF EQUATIONS                           *
C     *                                                         *
C     *   T       1-DIMENSION ARRAY, T(1) CONTAINS INITIAL TIME *
C     *                                                         *
C     *   TMAX    FINAL TIME                                    *
C     *                                                         *
C     *   VZERO   1-DIMENSIONAL ARRAY, CONTAINING INITIAL VALUES*
C     ***********************************************************
      COMMON A(10,10),T(4)
      DIMENSION Y(4,10),VZERO(10),VNEW(10)
      DOUBLE PRECISION A,ERR,H,HMAX,T,TMAX,V,VZERO,Z,HMIN
      DOUBLE PRECISION X,X1,X2,Z1,Z2,Z3,W1,W2,W3
      DOUBLE PRECISION HUSED,VNEW
      DATA ERR/.1D-8/,HMAX/1.D0/
      Z1=-0.1945338968080D-6
      Z2= 0.6110474831446D0
      Z3= 1.388950571618D0
      N=3
```

```
      KSTEP=2
      IAT=0
      IPC=0
      INDEX=0
      IGFN=1
      TMAX=48.D0
      HMAX=1.D0/8.D0
      H=1.D0/256.D0



      A(1,1)=-3520.D0
      A(1,2)=-0.013D0
      A(1,3)=0.D0
      A(2,1)=-1000.D0
      A(2,2)=-0.013D0
      A(2,3)=0.D0
      A(3,1)=-2500.D0
      A(3,2)=0.D0
      A(3,3)=-0.01D0
   10 H=4.D0*H
      T(1)=0.D0
      VZERO(1)=0.D0
      VZERO(2)=0.D0
      VZERO(3)=0.D0
      IF(H .GT. 16.D0) STOP
      CALL DIFEQ(ERR,H,HMAX,KSTEP,N,TMAX,V,VZERO,IGFN,IPC,
     2           INDEX,IAT,Z,VNEW,HUSED,HMIN)
      WRITE(5,1) HUSED,Z
    1 FORMAT(10X,'H      =',E15.8,10X,'T      =',E15.8//)
    2 FORMAT(1X,3(D24.16,2X))
    3 FORMAT(/15X,'MAX REL ERROR =',D24.16/)
      X1=VNEW(2)+1.D0
      X2=VNEW(3)+1.D0
      WRITE(5,2) VNEW(1),X1,X2
      WRITE(5,2) Z1,Z2,Z3
      U1=DABS((VNEW(1)-Z1)/Z1)
      U2=DABS((X1-Z2)/Z2)
      U2=DMAX1(U1,U2)
      U3=DABS((X2-Z3)/Z3)
      U3=DMAX1(U2,U3)
      WRITE(5,3) U3
      GO TO 10
      END
$


      SUBROUTINE START(KSTEP,H,N,Y,YN,YOLD,IT,IAT)
      COMMON A(10,10),T(10
      DIMENSION Y(4,10),YN(10),YOLD(4,10)
      DOUBLE PRECISION A,H,T,Y,YN,YOLD
      DATA I0,I1/0,1/
      DO 1 I=1,N
    1 YOLD(1,I)=Y(1,I)
      DO 10 J=1,KSTEP
      CALL NLAS(I1,H,YOLD,N,YN,I0,I1,IT,IAT,I0)
      DO 5 I=1,N
      YOLD(1,I)=YN(I)
    5 Y(J+1,I)=YN(I)
   10 CONTINUE
      RETURN
      END


      SUBROUTINE AFNT(A,N,T)
      DIMENSION A(10,10)
      DOUBLE PRECISION A,T
      RETURN
      END
```

```
      SUBROUTINE GEN(G,H,N,Y,J,T,A)
      DOUBLE PRECISION A(10,10),Y(4,10),G(10),T(4),H,TH
      G(2)=-0.013D0-1000.D0*Y(J,1)*Y(J,2)
      TH=-2500.D0*Y(J,1)*Y(J,3)
      G(1)=TH + G(2)
      G(3)=.1D-18Y(J,3) + TH
      RETURN
      END


      SUBROUTINE INVERT(A,M,ANS)
C     ***************************************************************************
C     *   MATRIX INVERSION SUBROUTINE, CALLED BY PADE OR NUMS                   *
C     *   A CONTAINS THE ORIGINAL ELEMENTS AND REMAINS UNALTERED                *
C     *   ANS CONTAINS THE A**(-1)                                              *
C     *   THIS SET-UP IS USING UNIVAC 1108 MATHPK EXISTING DOUBLE               *
C     *        PRECISION GAUSS-JORDAN REDUCTION                                 *
C     *   THIS PROGRAM IS REPLACEABLE BY THE USER                               *
C     ***************************************************************************
      DOUBLE PRECISION A(10,10),ANS(10,10),V(2)
      DIMENSION JC(10)
      DATA NR/10/,NC/10/
      V(1)=1.D0
      DO 1 I=1,M
      DO 1 J=1,M
    1 ANS(I,J)=A(I,J)
      CALL DGJR(ANS,NR,NC,M,N,MDEX,JC,V)
      IF(MDEX .EQ. 1) GO TO 10
      RETURN
   10 WRITE (4,2)
    2 FORMAT(3X,22HMATRIX INVERSION ERROR)
      RETURN
      END


      SUBROUTINE PADE(A,H,P,B,C,COL,N)
C     ***************************************************************************
C     *   CALCULATES MATRIX EXPONENTIAL BY PADE APPROXIMATION                   *
C     *   CALLED BY NUMS SUBROUTINE                                             *
C     ***************************************************************************
      DOUBLE PRECISION A(10,10),P(10,10),B(10,10),C(10,10),COL(10)
      DOUBLE PRECISION H,HAVE,XNORM
      HAVE=H
      DO 2 I=1,N
           DO 1 J=1,N
           B(I,J)=0.D0
           C(I,J)=0.D0
           P(I,J)=0.D0
    1      CONTINUE
      COL(I)=0.D0
    2 CONTINUE
      DO 17 I=1,N
      DO 16 J=1,N
      COL(I)=COL(I)+DABS(A(J,I))
   16 CONTINUE
   17 CONTINUE
      XNORM=COL(1)
      DO 18 I=1,N
      IF(XNORM .GT. COL(I)) GO TO 18
      XNORM=COL(I)
   18 CONTINUE
C     ***************************************************************************
C     *   COLUMN NORM IS USED TO SEE WHETHER EXP(A) NEEDS REDUCTION             *
C     ***************************************************************************
      M=0
   30 IF(XNORM-H - .1D0) 3,20,20
C     ***************************************************************************
C     *   EXP(A)=(I-0.5A+A**2/12.0)**(-1)*(I+0.5A+A**2/12.0)                    *
C     ***************************************************************************
    3 DO 6 I=1,N
           DO 5 J=1,N
               DO 4 K=1,N
               P(I,J)=P(I,J)+A(I,K)*A(K,J)
    4          CONTINUE
      C(I,J)=(P(I,J)*H*H/12.D0-A(I,J)/2.D0)*H
    5      CONTINUE
      C(I,I)=C(I,I)+1.0D0
```

```
    6 CONTINUE
      CALL INVERT(C,N,B)
      DO 9 I=1,N
           DO 10 J=1,N
      C(I,J)=(P(I,J)*H/12.D0+A(I,J)/2.D0)*H
           P(I,J)=0.D0
   10      CONTINUE
      C(I,I)=C(I,I)+1.0D0
    9 CONTINUE
      DO 12 I=1,N
      DO 13 J=1,N
      DO 14 K=1,N
      P(I,J)=P(I,J)+B(I,K)*C(K,J)
   14 CONTINUE
   13 CONTINUE
   12 CONTINUE
      IF(M .EQ. 0) GO TO 40
C     ***************************************************************
C     *   NORM(AH).GT.(.1), EXP(A) =EXP(A/2**M)**(2**M)             *
C     ***************************************************************



      DO 24 I=1,N
           DO 25 J=1,N
           B(I,J)=0.D0
   25      CONTINUE
   24 CONTINUE
      DO 36 K=1,M
      DO 27 I=1,N
      DO 28 J=1,N
      DO 29 L=1,N
      B(I,J)=B(I,J)+P(I,L)*P(L,J)
   29 CONTINUE
   28 CONTINUE
   27 CONTINUE
      DO 31 I=1,N
      DO 32 J=1,N
      P(I,J)=B(I,J)
      B(I,J)=0.D0
   32 CONTINUE
   31 CONTINUE
   36 CONTINUE
      H=HAVE
      RETURN
   20 H=H/2.D0
      R=R+1
      DO 54 I=1,N
      DO 55 J=1,N
      P(I,J)=0.D0
   55 CONTINUE
   54 CONTINUE
      GO TO 30
   40 H=HAVE
      RETURN
      END
*


      SUBROUTINE DIFEQ(ERR,H,HMAX,KSTEP,N,TMAX,Y,YZERO),IC,IV,
     *                 INDEX,IAT,ANORM,YNEW,MUSED,MMIN)
C     ***************************************************************
C     *   DIFEQ IS CALLED BY MAIN PROGRAM WHOSE ARGUMENTS ARE ALREADY *
C     *   DEFINED IN THE MAIN PROGRAM WHERE IC=ICFN, IV=IPC           *
C     *                                                               *
C     *   DIFEQ CALLS 2 SUBROUTINES                                   *
C     *         START(KSTEP,....,IAT) ---A STARTER                    *
C     *         NLMS (KSTEP,....,IAT) ---NONLINEAR MULTISTEP (GAB,GAM)*
C     *                                                               *
C     *   SOLUTION VECTOR Y(T) IS YNEW(I) OR Y(KSTEP+1,I)             *
C     *                                                               *
C     *   BASED ON USER-SUPPLIED INPUTS, DIFEQ SETS UP THE ITERATIVE  *
C     *   PROCEDURE, CONTROLS STARTER, STEP-SIZE CHANGES, PREDICTOR-  *
C     *   CORRECTOR, CORRECTOR'S CONVERGENCE, CALLS FOR               *
C     *   THE REQUIRED METHODS                                        *
C     ***************************************************************
```

```
      COMMON A(10,10),T(4)
      DIMENSION G(10),Y(4,10),YN(10),YNEW(10),YOLD(4,10),YZERO(10)
      DOUBLE PRECISION A,ANORM,ERR,G,H,HMAX,HMIN,T,TEA,TMAX
      DOUBLE PRECISION TZERO,Y,YN,YNEW,YOLD,YZERO,HUSED
      DATA IZERO,IONE/0,1/
      LMT=3
      JH=0
      HUSED=H
      DO 48 I=1,N
   48 Y(1,I)=YZERO(I)
      ITER=0
      ISTEP=0
      TZERO=T(1)
   49 ITER=ITER+1
      IH=0
      IMIN=0
   50 CONTINUE
C     ****************************************************************
C     *   EVALUATE A(T) AT T=T(0) IF IAT NOT 0                       *
C     *   IF KSTEP GT 1, CALLS START                                 *
C     ****************************************************************
      IF(IAT.NE.0) CALL AFNT(A,N,T(1))
   51 IF(KSTEP.EQ.1 .AND. INDEX.EQ.0) GO TO 60
      CALL START(KSTEP,H,N,Y,YN,YOLD,IG,IAT)
   60 TEA=T(1)+DBLE(FLOAT(KSTEP))*H
      IF(TEA.GT.TMAX) H=(TMAX-T(1))/DBLE(FLOAT(KSTEP))
      IF(TEA.GT.TMAX) GO TO 51
      IH=IH+1
      IF(IH .GE. 32767) IH=2
      DO 61 J=1,KSTEP
   61 T(J+1)=T(J)+H
      IMP=KSTEP+1
      DO 62 J=1,IMP
      DO 62 I=1,N
   62 YOLD(J,I)=Y(J,I)
C     ****************************************************************
C     *   FIXED-STEP-SIZE      IV=0,   INDEX=0   PREDICTOR           *
C     *                        IV=0,   INDEX=1   CORRECTOR           *
C     *   VARIABLE-STEP-SIZE   IV NE 0,          PREDICTOR-CORRECTOR *
C     ****************************************************************
      IF(IV.EQ.0.AND.INDEX.EQ.0) CALL NLMS(KSTEP,H,YOLD,N,YN,IZERO,
     $                                     IH,IG,IAT,JH)
      IF(IV.EQ.0.AND.INDEX.NE.0) CALL NLMS(KSTEP,H,YOLD,N,YN,IONE,
     $                                     IH,IG,IAT,JH)
      IF(IV.NE.0) CALL NLMS(KSTEP,H,YOLD,N,YN,IZERO,IONE,IG,IAT,JH)
   65 DO 66 I=1,N



   66 YOLD(KSTEP+1,I)=YN(I)
      IF(IV.NE.0) GO TO 69
      IF(LMT .EQ. 1) GO TO 69
      DO 63 I=1,N
   68 YNEW(I)=YN(I)
      GO TO 82
C     ****************************************************************
C     *   CORRECTOR AT MOST CORRECT 3 TIMES                          *
C     *   STEP-SIZE CHANGED BY A FACTOR OF 2                         *
C     ****************************************************************
   69 ICORR=0
   70 ICORR=ICORR+1
      IF(ICORR .LE. LMT) GO TO 75
      H=H/2.D0
      JH=0
      IF(H .LT. HMIN) H=HMIN
      IF(H .LT. HMIN) IMIN=IMIN+1
      IF(IMIN .GT. 3) WRITE (6,1170)
 1170 FORMAT(3X,39HH REACHED HMIN, NO CONVERGENCE POSSIBLE)
      IF(IMIN .GT. 3) STOP
      T(1)=TZERO
      TEA=T(1)
      DO 71 I=1,N
   71 Y(1,I)=YZERO(I)
      GO TO 50
   75 CALL NLMS(KSTEP,H,YOLD,N,YNEW,IONE,ICORR,IG,IAT,JH)
      IF(LMT .NE. 1) GO TO 76
      GO TO 82
   76 CONTINUE
      ANORM=0.D0
```

```
C     ***************************************************************
C     *   TEST CORRECTOR'S CONVERGENCE, USING MAX NORM              *
C     ***************************************************************
      DO 80 J=1,N
      G(J)=(YOLD(KSTEP+1,J)-YNEW(J))
      IF(ANORM .GT. DABS(G(J))) GO TO 80
      ANORM=DABS(G(J))
   80 CONTINUE
      IF(ANORM-ERR) 82,82,1182
 1182 DO 81 J=1,N
   81 YOLD(KSTEP+1,J)=YNEW(J)
      GO TO 70
   82 DO 83 I=1,N
   83 Y(KSTEP+1,I)=YNEW(I)
C     ***************************************************************
C     *   RESULTS Y(TEA) IN YNEW(I) AND Y(KSTEP+1,I)                *
C     ***************************************************************
      ANORM=TEA
      MUSED=DMAX1(M,MUSED)
      DO 85 J=1,KSTEP
      DO 85 I=1,N
      Y(J,I)=Y(J+1,I)
      IF(J.EQ.1) YZERO(I)=Y(J,I)
   85 CONTINUE
      JM=1
      T(1)=T(2)
      TZERO=T(1)
      IF(IV.EQ.0 .AND. INDEX.EQ.1) LMT=1
      IF(IV.EQ.0 .AND. INDEX.EQ.1) INDEX=0
      IF(LMT .EQ. 1) IH=0
      IF(DABS(TEA-TMAX) .LE. (.1D-7)) RETURN
      IF(IV .EQ. 0) GO TO 60
      T(1)=TEA



      TZERO=T(1)
      DO 86 I=1,N
      YZERO(I)=Y(KSTEP+1,I)
   86 Y(1,I)=Y(KSTEP+1,I)
C     ***************************************************************
C     *   MAINTAIN SUCCESSFLL H CONSTANTLY FOR 3 TIMES BEFORE DOUBLING *
C     ***************************************************************
      ISTEP=ISTEP+1
      IF(ISTEP .LT. 3) GO TO 49
      ISTEP=0
      H=2.D0*H
      IF(H .GT. HMAX) H=HMAX
      IF(DABS(TEA-TMAX) .LE. (.1D-7)) RETURN
      GO TO 49
      END
C

      SUBROUTINE NLMS(KSTEP,M,Y,N,YN,INDEX,IS,IT,IAT,JM)
C     ***************************************************************
C     *   NONLINEAR MULTISTEP ALGORITHM(GAB,GAM)                    *
C     *         CALLED BY DIFEQ OR START                            *
C     *   ARGUMENTS ALREADY DEFINED IN MAIN PROGRAM                 *
C     *   THIS PROGRAM CALLS 3 SUBROUTINES                          *
C     *         INVERT(AH,N,P1)                                     *
C     *         GFN(G,....,A)                                       *
C     *         PADE(A,....,N)                                      *
C     *   SOLUTION VECTOR IS STORED IN YN(I)                        *
C     ***************************************************************
      COMMON A(10,10),T(4)
      DIMENSION AH(10,10),AH2(10,10),AH3(10,10),AH4(10,10),AT(10,10)
      DIMENSION EAH(10,10),E2AH(10,10),E3AH(10,10),G(10)
      DIMENSION P1(10,10),PHI(4,10,10),QHI(4,10,10),UNIT(10,10)
      DIMENSION U1(2,10,10),U2(2,10,10),U3(2,10,10),U4(2,10,10)
      DIMENSION Y(4,10),YN(10)
      DIMENSION FE(3,10,10),F1(4,10,10),A1(10,10),A2(10,10),A3(10,10)
      DOUBLE PRECISION A,AH,AH2,AH3,AH4,AT,EAH,E2AH,E3AH,G,H
      DOUBLE PRECISION P1,PHI,QHI,T,T1,T2,T3,TH,UNIT,U1,U2,U3,U4,Y,YN
      DOUBLE PRECISION FE,F1,A1,A2,A3
C     ***************************************************************
C     *   JM AND IS ARE INDICATORS                                  *
C     *   PROGRAM DOES INITIALIZATION WHEN IS=1 AND JM=0            *
C     *   PROGRAM CALCULATES AND SAVES AH, EXP(AH), A INVERSE,PHI FN *
C     *   IS.GT.1 ABOVE CALCULATIONS ARE BYPASSED                   *
C     ***************************************************************
```

```
      IF(JH .GT. 0) GO TO (100,200,300), KSTEP
      DO 2 I=1,N
           DO 2 J=1,N
           A2(I,J)=0.D0
           A3(I,J)=0.D0
           AH(I,J)=H*A(I,J)
           AH2(I,J)=0.D0
           AH3(I,J)=0.D0
    2      AH4(I,J)=0.D0
      IF(N-1) 7,8,7
    8 A1(1,1)=1.D0/AH(1,1)
      A2(1,1)=A1(1,1)*A1(1,1)
      AH2(1,1)=AH(1,1)*AH(1,1)
      A3(1,1)=A2(1,1)*A1(1,1)
      AH3(1,1)=AH2(1,1)*AH(1,1)
      GO TO 9
    7 CALL INVERT(AH,N,A1)
      DO 21 I=1,N
      DO 21 J=1,N
      DO 21 K=1,N
      AH2(I,J)=AH2(I,J)+AH(I,K)*AH(K,J)
   21 A2(I,J)=A2(I,J)+A1(I,K)*A1(K,J)
      IF(KSTEP .EQ. 1) GO TO 9
      DO 22 I=1,N
      DO 22 J=1,N
      DO 22 K=1,N
      AH3(I,J)=AH3(I,J)+AH2(I,K)*AH(K,J)
   22 A3(I,J)=A3(I,J)+A2(I,K)*A1(K,J)
      IF(KSTEP .EQ. 2) GO TO 9
      DO 23 I=1,N
      DO 23 J=1,N
      DO 23 K=1,N
   23 AH4(I,J)=AH4(I,J)+A3(I,K)*A1(K,J)
    9 IF(N-1) 10,11,10
   11 EAH(1,1)=DEXP(AH(1,1))

      GO TO 14
   10 CALL PADE(A,H,EAH,E2AH,E3AH,C,N)
   14 IF(IS .GT. 1) GO TO (100,200,300), KSTEP
      DO 1 I=1,N
           DO 6 J=1,N
           FE(1,I,J)=0.D0
           FE(2,I,J)=0.D0
           FE(3,I,J)=0.D0
           PI(I,J)=0.D0
           UNIT(I,J)=0.D0
           E2AH(I,J)=0.D0
           E3AH(I,J)=0.D0
    6      CONTINUE
    1 UNIT(I,I)=1.D0
      DO 3 I=1,4
      DO 3 J=1,N
      DO 3 K=1,N
      FI(I,J,K)=0.D0
      PHI(I,J,K)=0.D0
    3 GHI(I,J,K)=0.D0
      GO TO (100,200,300) , KSTEP
  100 CONTINUE
C     ***************************************************************************
C     *   NONLINEAR MULTISTEP STARTS HERE.                                      *
C     *   BEGINNING SECTION DOES INITIALIZATION                                 *
C     ***************************************************************************
      DO 132 I=1,N
      YN(I)=0.D0
      PHI(2,I,1)=0.D0
  132 PHI(3,I,1)=0.D0
      IF(IS.GT.1 .AND. INDEX.EQ.0) GO TO 131
      DO 103 I=1,N
      DO 103 J=1,N
  103 PI(I,J)=-EAH(I,J)+UNIT(I,J)
C     ***************************************************************************
C     *   1ST ORDER GAB                                                         *
C     *   DO LOOP 105 CALCULATES PHI(1,0)                                       *
C     *   LOOP 108 OR 110 COMPUTES FINAL Y(N+1)                                 *
C     ***************************************************************************
      IF(INDEX .NE. 0) GO TO 153
  104 IF(IT .EQ. 0) GO TO 109
      DO 106 I=1,N
      DO 106 J=1,N
      DO 105 K=1,N
```

```
      105 PHI(1,I,J)=PHI(1,I,J)-A1(I,K)*P1(K,J)
          FE(1,I,J)=PHI(1,I,J)
      106 CONTINUE
      131 CALL GFN(G,H,N,V,KSTEP,T,A)
          DO 108 I=1,N
          DO 108 J=1,N
      108 YN(I)=YN(I)+EAH(I,J)*V(1,J)+H*FE(1,I,J)*G(J)
          GO TO 172
      109 DO 110 I=1,N
          DO 110 J=1,N
      110 YN(I)=YN(I)+EAH(I,J)*V(1,J)
          GO TO 172
C     ***********************************************************************
C     *   1ST ORDER GAM                                                     *
C     *   COMPUTE PHI(1,0), PHI(1,1)                                        *
C     *   FINAL RESULTS ARE CALCULATED IN LOOP 166 OR 168                   *
C     ***********************************************************************
      153 IF(IT .EQ. 0) GO TO 167
          DO 160 I=1,N


              DO 161 J=1,N
                  DO 162 K=1,N
      162         FI(1,I,J)=FI(1,I,J)-H*A(I,K)*EAH(K,J)
              FI(1,I,J)=FI(1,I,J)-P1(I,J)
      161     FI(2,I,J)=P1(I,J)+H*A(I,J)
      160 CONTINUE
      170 K2=KSTEP+1
          IF(IT .EQ. 0) GO TO 167
          DO 163 I=1,N
              DO 164 K=1,K2
              CALL GFN(G,H,N,V,K,T,A)
                  DO 165 J=1,N
                  IF(K.EQ.1) PHI(3,I,1)=PHI(3,I,1)+FI(1,I,J)*G(J)
      165         IF(K.EQ.2) PHI(2,I,1)=PHI(2,I,1)+FI(2,I,J)*G(J)
      164     PHI(3,I,1)=PHI(3,I,1)+PHI(2,I,1)
      163 CONTINUE
          DO 166 I=1,N
          DO 166 K=1,N
      166 YN(I)=YN(I)-H*A2(I,K)*PHI(3,K,1)+EAH(I,K)*V(1,K)
          GO TO 172
      167 DO 168 I=1,N
          DO 168 K=1,N
      168 YN(I)=YN(I)+EAH(1,K)*V(1,K)
C     ***********************************************************************
C     *   IF A IS A FUNCTION OF T, PERFORM PERIODIC DECOMPOSITION,          *
C     *   EVALUATE A(T(I)), AND ADJUST FINAL Y(N+1)                         *
C     ***********************************************************************
      172 IF(IAT .EQ. 0) RETURN
          DO 125 I=1,N
          DO 125 J=1,N
      125 E2AH(I,J)=FI(2,I,J)
      176 TH=T(1)+H
          CALL AFNT(AT,N,TH)
          DO 173 I=1,N
          G(I)=0.D0
          DO 173 J=1,N
      173 G(I)=G(I)+(AT(I,J)-A(I,J))*V(2,J)
          DO 174 I=1,N
          QHI(4,4,I)=0.D0
          DO 174 J=1,N
      174 QHI(4,4,I)=QHI(4,4,I)+E2AH(I,J)*G(J)
          DO 175 I=1,N
          DO 175 J=1,N
      175 YN(I)=YN(I)-H*A2(I,J)*QHI(4,4,J)
          RETURN
      200 CONTINUE
C     ***********************************************************************
C     *   NONLINEAR MULTI-2-STEP STARTS HERE                                *
C     *   BEGINNING SECTION DOES INITIALIZATION                             *
C     ***********************************************************************
          DO 240 I=1,N
              DO 241 J=1,N
              DO 241 L=1,4
              PHI(L,J,I)=0.D0
      241     P1(I,J)=0.D0
      240 YN(I)=0.D0
          IF(INDEX .GT. 0) GO TO 260
```

```
C     ***************************************************************************
C     *   2ND ORDER GAB                                                         *
C     *   A2 CONTAINS (AH)**(-2)                                                *
C     ***************************************************************************
      IF(IS .GT. 1) GO TO 234

      IF(IT .EQ. 0) GO TO 235
C     ***************************************************************************
C     *   AT THE FINISH OF LOOP 215 --                                          *
C     *   PHI(1,I,J) CONTAINS PHI(1,0), PHI(2,I,J) CONTAINS PHI(2,0)            *
C     *   FINAL Y(N+1) ARE CALCULATED IN LOOP 232                               *
C     ***************************************************************************
      DO 213 I=1,N
      DO 213 J=1,N
  213 P1(I,J)=-EAH(I,J)+UNIT(I,J)
      DO 215 I=1,N
      DO 215 J=1,N
      DO 215 K=1,N
  215 PHI(2,I,J)=PHI(2,I,J)-AH(I,K)*EAH(K,J)
      DO 218 I=1,N
           DO 219 J=1,N
           PHI(1,I,J)=PHI(1,I,J)-P1(I,J)-AH(I,J)
           FE(1,I,J)=PHI(1,I,J)
           PHI(2,I,J)=PHI(2,I,J)+P1(I,J)+2.0D0*AH(I,J)
  219      FE(2,I,J)=PHI(2,I,J)
  218 P1(1,I)=0.D0
  234 DO 206 I=1,N
      DO 206 J=1,N
      PHI(1,I,J)=FE(1,I,J)
  206 PHI(2,I,J)=FE(2,I,J)
      DO 220 K=1,KSTEP
      CALL GFN(G,H,N,V,K,T,A)
      DO 221 I=1,N
      DO 221 J=1,N
  221 YN(I)=YN(I)+PHI(K,I,J)*G(J)*H
  220 CONTINUE
      DO 230 I=1,N
      DO 230 J=1,N
  230 P1(1,I)=P1(1,I)-A2(I,J)*YN(J)
  235 DO 232 I=1,N
      IF(IT .EQ. 0) P1(1,I)=0.D0
      DO 233 J=1,N
  233 PHI(4,1,I)=PHI(4,1,I)+EAH(I,J)*V(2,J)
  232 YN(I)=P1(1,I)+PHI(4,1,I)
      IF(IAT .EQ. 0) RETURN
      DO 290 I=1,N
      DO 290 J=1,N
  290 E2AH(I,J)=PHI(2,I,J)
      GO TO 176
C     ***************************************************************************
C     *   2ND ORDER GAM                                                         *
C     *   NEXT BELOW 267,  A3 CONTAINS (AH)**(-3)                               *
C     ***************************************************************************
  280 CONTINUE
      IF(IS .GT. 1) GO TO 278
      IF(IT .EQ. 0) GO TO 285
C     ***************************************************************************
C     *   END OF LOOP 275, PHI(L,I,J) CONTAINS PHI(2,L),L=0,1,2                 *
C     *   FINAL Y(N+1) ARE CALCULATED IN LOOP 282 OR 286                        *
C     ***************************************************************************
      DO 270 I=1,N
      DO 270 J=1,N
      U1(1,I,J)=UNIT(I,J)-0.5D0*AH(I,J)
      U1(2,I,J)=UNIT(I,J)+0.5D0*AH(I,J)
      U2(1,I,J)=-2.D0*UNIT(I,J)+AH2(I,J)
      U2(2,I,J)=-2.D0*(UNIT(I,J)+AH(I,J))
      U3(1,I,J)=U1(2,1,J)
  270 U3(2,I,J)=UNIT(I,J)+1.5D0*AH(I,J)+AH2(I,J)
      DO 272 I=1,N

           DO 273 J=1,N
               DO 274 L=1,N
               QHI(1,I,J)=QHI(1,I,J)-U1(1,I,L)*EAH(L,J)
               QHI(2,I,J)=QHI(2,I,J)-U2(1,I,L)*EAH(L,J)
               QHI(3,I,J)=QHI(3,I,J)-U3(1,I,L)*EAH(L,J)
  274          CONTINUE
           QHI(1,I,J)=QHI(1,I,J)+U1(2,I,J)
           QHI(2,I,J)=QHI(2,I,J)+U2(2,I,J)
           QHI(3,I,J)=QHI(3,I,J)+U3(2,I,J)
```

```
  273      CONTINUE
  272 CONTINUE
      DO 275 L=1,3
      DO 275 I=1,N
      DO 275 J=1,N
      DO 275 K=1,N
      PHI(L,I,J)=PHI(L,I,J)-A3(I,K)*PHI(L,K,J)
  275 FI(L,I,J)=PHI(L,I,J)
  279 DO 288 L=1,3
      DO 288 I=1,N
      DO 288 J=1,N
  288 PHI(L,I,J)=FI(L,I,J)
      DO 280 I=1,N
      DO 280 K=1,3
      CALL GFM(G,M,N,Y,K,T,A)
      DO 282 J=1,N
      YN(I)=YN(I)+PHI(K,I,J)*G(J)*H
  282 IF(K.EQ.3) YN(I)=YN(I)+EAH(I,J)*Y(2,J)
  280 CONTINUE
      GO TO 293
  285 DO 286 I=1,N
      DO 286 J=1,N
  286 YN(I)=YN(I)+EAH(I,J)*Y(2,J)
  293 IF(IAT .EQ. 0) RETURN
C     ************************************************************************
C     *   IF A IS A FUNCTION OF T, PERFORM PERIODIC DECOMPOSITION          *
C     *   EVALUATE A(T(I)), AND ADJUST FINAL Y(N+1)                        *
C     ************************************************************************
  287 T1=T(1)+H
      T2=T1+H
      CALL AFNT(E3AH,N,T1)
      CALL AFNT(AT,N,T2)
      DO 294 I=1,N
      G(I)=0.D0
      P1(1,I)=0.D0
      DO 294 J=1,N
      G(I)=G(I)+(E3AH(I,J)-A(I,J))*Y(2,J)
  294 P1(1,I)=P1(1,I)+(AT(I,J)-A(I,J))*Y(3,J)
      DO 295 I=1,N
      P1(2,I)=0.D0
      DO 295 J=1,N
  295 P1(2,I)=P1(2,I)+PHI(2,I,J)*G(J)+PHI(3,I,J)*P1(1,J)
      IF(KSTEP .EQ. 3) GO TO 298
      DO 296 I=1,N
  296 YN(I)=YN(I)+H*P1(2,I)
      RETURN
  298 DO 299 I=1,N
  299 YN(I)=YN(I)-H*P1(2,I)
      RETURN
  300 CONTINUE
C     ************************************************************************
C     *   3RD ORDER GAB & GAM                                                *
C     *   BEGINNING SECTION DOES INITIALIZATION                              *
C     *   BEFORE 303, THE FOLLOWING RESULTS ARE STORED...                    *

C     *   EAH=EXP(AH),  A3=(AH)**(-3)                                        *
C     ************************************************************************
      KUP=KSTEP
      IF(INDEX .EQ. 1) KUP=KSTEP+1
      DO 320 I=1,N
  320 YN(I)=0.D0
      IF(IS .GT. 1) GO TO 321
      IF(INDEX .EQ. 1) GO TO 357
  303 IF(IT .EQ. 0) GO TO 370
C     ************************************************************************
C     *   CALCULATE PHI(3,K), K=0,1,2. RESULTS IN PHI(K,I,J)                 *
C     ************************************************************************
      DO 304 I=1,N
      DO 304 J=1,N
      U1(1,I,J)=UNIT(I,J)+0.5D0*AH(I,J)
      U1(2,I,J)=UNIT(I,J)+1.5D0*AH(I,J)+AH2(I,J)
      U2(1,I,J)=-2.D0*(UNIT(I,J)+AH(I,J))
      U2(2,I,J)=-2.D0*UNIT(I,J)-4.D0*AH(I,J)-3.D0*AH2(I,J)
      U3(1,I,J)=U1(2,I,J)
  304 U3(2,I,J)=UNIT(I,J)+2.5D0*AH(I,J)+3.D0*AH2(I,J)
```

```
      DO 307 I=1,N
        DO 308 J=1,N
          DO 309 K=1,N
          GMI(1,I,J)=GMI(1,I,J)-U1(1,I,K)*EAN(K,J)
          GMI(2,I,J)=GMI(2,I,J)-U2(1,I,K)*EAN(K,J)
          GMI(3,I,J)=GMI(3,I,J)-U3(1,I,K)*EAN(K,J)
          IF(KUP .NE. 4) GO TO 309
          GMI(4,I,J)=GMI(4,I,J)-U4(1,I,K)*EAN(K,J)
  309     CONTINUE
        GMI(1,I,J)=GMI(1,I,J)+W1(2,I,J)
        GMI(2,I,J)=GMI(2,I,J)+W2(2,I,J)
        GMI(3,I,J)=GMI(3,I,J)+W3(2,I,J)
        IF(KUP.EQ.4) GMI(4,I,J)=GMI(4,I,J)+W4(2,I,J)
  308   CONTINUE
  307 CONTINUE
      DO 310 K=1,KUP
      DO 310 I=1,N
      DO 310 J=1,N
      DO 310 L=1,N
      IF(INDEX.EQ.0) PHI(K,I,J)=PHI(K,I,J)- A3(I,L)*GMI(K,L,J)
      IF(INDEX.EQ.0) FE(K,I,J)=PHI(K,I,J)
      IF(INDEX.EQ.1) PHI(K,I,J)=PHI(K,I,J)-AM4(I,L)*GMI(K,L,J)
  310 IF(INDEX.EQ.1) FI(K,I,J)=PHI(K,I,J)
  321 DO 311 K=1,KUP
      DO 311 I=1,N
      DO 311 J=1,N
      IF(INDEX.EQ.0) PHI(K,I,J)=FE(K,I,J)
  311 IF(INDEX.EQ.1) PHI(K,I,J)=FI(K,I,J)
      DO 314 K=1,KUP
      CALL GFN(G,H,N,V,K,T,A)
      DO 316 I=1,N
      DO 316 J=1,N
C     ************************************************************
C     *   CALCULATE FINAL Y(N+1)                                 *
C     ************************************************************
      VN(I)=VN(I)+HSPHI(K,I,J)*G(J)
  316 IF(K.EQ.KUP) VN(I)=VN(I)+EAN(I,J)*BV(3,J)
  314 CONTINUE
      GO TO 340
  370 DO 371 I=1,N
      DO 371 J=1,N
  371 VN(I)=VN(I)+EAN(I,J)*BV(3,J)
  340 IF(IAT .EQ. 0) RETURN


      IF(INDEX .EQ. 0) GO TO 397
C     ************************************************************
C     *   TO SEE IS A A FUNCTION OF T                            *
C     ************************************************************
      T1=T(1)+H
      T2=T1+H
      T3=T2+H
      CALL AFNT(AT,N,T1)
      CALL AFNT(EAN,N,T2)
      CALL AFNT(P1,N,T3)
      DO 341 I=1,N
      G(I)=0.D0
      GMI(4,3,I)=0.D0
      GMI(4,4,I)=0.D0
      DO 341 J=1,N
      G(I)=G(I)+(AT(I,J)-A(I,J))*BV(2,J)
      GMI(4,3,I)=GMI(4,3,I)+(EAN(I,J)-A(I,J))*BV(3,J)
  341 GMI(4,4,I)=GMI(4,4,I)+(P1(I,J)-A(I,J))*BV(4,J)
      DO 342 I=1,N
      DO 342 J=1,N
  342 VN(I)=VN(I)+H*(PHI(2,I,J)*G(J)+PHI(3,I,J)*GMI(4,3,J)+PHI(4,I,J)*GM
     8I(4,4,J))
      RETURN
C     ************************************************************
C     *   CALCULATE IMPLICIT PHI FUNCTION, PHI(J,J), J=0,1,2,3   *
C     ************************************************************
  397 IF(IT .EQ. 0) GO TO 370
      DO 358 I=1,N
      DO 358 J=1,N
      U1(1,I,J)=-UNIT(I,J)+AM2(I,J)/6.D0
      U1(2,I,J)=-UNIT(I,J)-AM(I,J)-AM2(I,J)/3.D0
      W2(1,I,J)=3.D0*UNIT(I,J)+AM(I,J)-AM2(I,J)
```

```
      U2(2,I,J)=3.D0*UNIT(I,J)+4.D0*AN(I,J)+1.5D0*AN2(I,J)
      U3(1,I,J)=-3.D0*UNIT(I,J)-8.D0*AN(I,J)+0.5D0*AN2(I,J)+AN3(I,J)
      U3(2,I,J)=-3.D0*UNIT(I,J)-6.D0*AN(I,J)-3.D0*AN2(I,J)
      U4(1,I,J)=-U1(2,I,J)
  300 U4(2,I,J)=UNIT(I,J)+8.D0*AN(I,J)+11.D0*AN2(I,J)/6.D0+AN3(I,J)
      GO TO 300
      END
$
```

```
RUN CAPM1
        H     = 0.15625000E-01          T     = 0.48000000E+02

 -0.19453396019387993D-05    0.61104746395342450D+00    0.13889504784953830D+01
 -0.19453369680800000D-05    0.61104748314460000D+00    0.13889505715160000D+01

             MAX REL ERROR = 0.66971869849989880D-07

        H     = 0.62500000E-01          T     = 0.48000000E+02

 -0.19453387748433630D-05    0.61104750862073660D+00    0.13889503865174500D+01
 -0.19453389680800000D-05    0.61104748314460000D+00    0.13889505715160000D+01

             MAX REL ERROR = 0.13319304054340020D-06

        H     = 0.25000000E+00          T     = 0.48000000E+02

 -0.19453358496762720D-05    0.61104795588261670D+00    0.13889478538524710D+01
 -0.19453389680800000D-05    0.61104748314460000D+00    0.13889505715160000D+01

             MAX REL ERROR = 0.19566308437424240D-05

        H     = 0.10000000E+01          T     = 0.48000000E+02

 -0.19452864225386520D-05    0.61105505348426790D+00    0.13889069215790450D+01
 -0.19453389680800000D-05    0.61104748314460000D+00    0.13889505715160000D+01

             MAX REL ERROR = 0.31426558691302540D-04

        H     = 0.40000000E+01          T     = 0.48000000E+02

 -0.19443109925876030D-05    0.61116785338588360D+00    0.13882422214021780D+01
 -0.19453389680800000D-05    0.61104748314460000D+00    0.13889505715160000D+01

             MAX REL ERROR = 0.52843001104916800D-03

        H     = 0.16000000E+02          T     = 0.48000000E+02

 -0.19278812560754310D-05    0.61296919213510840D+00    0.13768156887550700D+01
 -0.19453389680800000D-05    0.61104748314460000D+00    0.13889505715160000D+01

             MAX REL ERROR = 0.89741834463623690D-02

TT4  --  STOP
```

INITIAL DISTRIBUTION LIST

Addressee	No. of Copies
ONR, ONR-100, -480	2
CNM, MAT-08T2, -08T1, -08T24	3
NRL	1
NORDA	1
NAVELECSYSCOM, ELEX 03, 320	2
NAVSEASYSCOM, SEA-03C, -63R, 63R-23, 09G32(4)	7
ARL/PENN STATE, STATE COLLEGE (Dr. S. McDaniel)	1
ARL, UNIV of Texas (R. Hawker)	1
DDC, ALEXANDRIA	12
Univ. of Illinois at Urbana-Champaign (Prof. C. W. Gear), Dept. of Computer Science, Urbana, IL 61801	1
Univ. of Wisconsin in-Madison, (Prof. W. E. Stewart), Chemical Engineering Dept., 1415 Johnson Drive, Madison, WI 53706	1
U.S. Army Ballistic Research Lab., (Dr. C. K. Zoltani), Bldg. 394, Aberdeen Proving Ground, MD	1
Harvard University, (Prof. Donald C. M. Anderson), The Aiken Computation Laboratory, Division of Applied Science, Cambridge, 02138	1
Virginia Commonwealth Univ., (Prof. J. Rosenbaum), 901 West Franklin Street, Richmond, VA 23284	1
Univ. of Tenn., (Mr. Bill Layton), Department of Mathematics, Knoxville, TN 37921	1
Univ. of Wisconsin, (Prof. T. Y. Li), MRC, Madison, WI 53706	1
Univ. of Helsinki, (Prof. Matti Makela), Dept. of Computer Science, Toolonkatu 11, SF-00100 Helsinki 10, Finland	1
NRL, Code 5309, (Mr. Harold L. Toothman), Washington, DC 20375	1
General Electric Co., (Mary Maier), Lamp Materials Research Laboratory, Bldg. 336, Nela Park, Cleveland, OH 44112	1
Dr. C. A. Steele, Jr., P.O. Box 45, Magnolia, MA 01930	1
Polytechnic Institute of New York, (Prof. S. Preiser) Mathematics Dept., 333 Jay Street, Brooklyn, NY 11201	1
Con Edison Co. of New York, Inc., (Philip Hsiang), 4 Irving Place, New York, NY 10003	1
TASC, (Dr. Joseph D'Appolito, Dr. Thomas O. Mottl, Dr. Sen Lee, Dr. R. Tracy) 6 Jacob Way, Reading, MA 01867	4
NUSC, (Dr. R. M. Kennedy), New London Lab Representative, 1651 Southwest 14th St., Ft. Lauderdale, FL 33315	1
Univ. of Rhode Island, (Prof. J. S. Papadakis), Dept. of Mathematics, Kingston, RI 02881	1
Univ. of Illinois at Chicago Circle, (Prof. R. W. Golland), Computer Center, Box 4348, Chicago, IL 60680	1
Univ. of Miami, (Prof. F. D. Tappert), Dept. of Mathematics, Coral Gables, FL 33124	1

☆ U.S. GOVERNMENT PRINTING OFFICE: 1979 — 614-006/25